Space Pirate
解説

問題概要

- 関数グラフfがある
- 頂点Aの行き先をf(A)から頂点Bに置き換える
- スタート地点(頂点1)からK歩進む
- (A,B)を色々動かしたときに、K歩進んだ先はどのようになるでしょうか。数えてください。

小課題1 (10点)

- $N \leq 100$

小課題1 (10点)

- $N \leq 100$
- 全ての(A, B)を試し、それぞれの行き先を調べる

小課題1 (10点)

- $N \leq 100$
- 全ての(A, B)を試し、それぞれの行き先を調べる
- Kが大きいのでループを検出してうまいこと処理する

小課題1,2 (10+37点)

- $N \leq 3000$

小課題1,2 (10+37点)

- $N \leq 3000$
- 何かしらのオーダーを減らせばよい

小課題1,2 (10+37点)

- $N \leq 3000$
- 各(A,B)について$O(\log n)$で調べられればよい

小課題1,2 (10+37点) – 考察

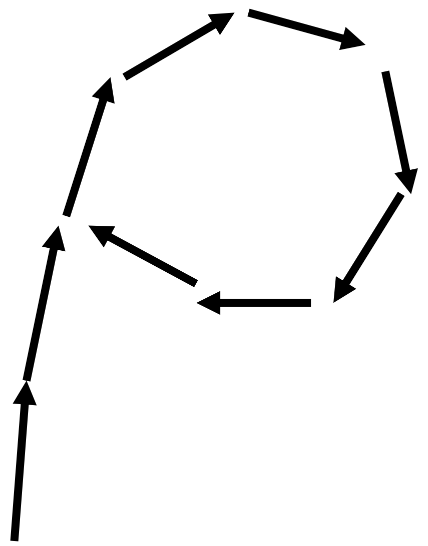
- $N \leq K$

小課題1,2 (10+37点) – 考察

- $N \leq K \rightarrow K$歩進んだ先は閉路上にある

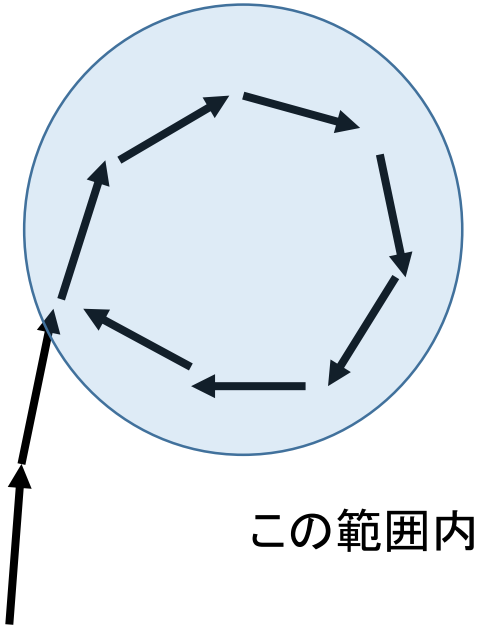
小課題1,2 (10+37点) – 考察

- $N \leq K \rightarrow K$歩進んだ先は閉路上にある



小課題1,2 (10+37点) – 考察

- $N \leq K \rightarrow K$歩進んだ先は閉路上にある



小課題1,2 (10+37点) – 事前計算

- 以下を事前計算

小課題1,2 (10+37点) – 事前計算

- 以下を事前計算
  - 元のグラフで、各点から2^i進んだ先はどの点か？ (doubling)
  - 各連結成分について根を一つ決めておく
    - 正確には木になっていないが、一箇所を切り開いて木と見なす
  - 各点について、その点が所属する連結成分の根
  - 各連結成分について、その連結成分に存在する閉路の大きさ

小課題1,2 (10+37点) – 事前計算

- doublingにより、ある点から元のグラフ上でX歩進んだ先は高速に求められる

小課題1,2 (10+37点) – 場合分け

- 以下のように場合分け

小課題1,2 (10+37点) – 場合分け

- 以下のように場合分け
  - 頂点1から頂点Aに到達できない場合
    →全て同じ点に到達する。到達先は簡単にわかる
  - 頂点1から頂点Aに到達可能な場合
    →さらに場合分け

小課題1,2 (10+37点) – 場合分け

- 頂点1から頂点Aに到達可能な場合
  - 閉路上の到達可能な点を一つ選び、Cとおく
  - 頂点1からCに初到達するまでの歩数を計算
  - Cからの残り歩数を閉路長で割り、余りをとる
  - この余りの数だけ進んだ点が、到達地点

小課題1,2 (10+37点) – 場合分け

- 以下のように場合分け
  - 頂点Bが頂点Aの子孫である場合
    →もとの閉路とは無関係の閉路ができる (C=Bとおく)
  - 頂点Bが頂点Aの子孫ではなく、頂点Aがもとの閉路上にある場合
    →もとの閉路の一部が変更されたような閉路ができる (C=Bとおく)
  - 頂点Bが頂点Aの子孫ではなく、頂点Aがもとの閉路上にない場合
    →もとの閉路が使われる

小課題1,2 (10+37点) – 場合分け

- 以下のように場合分け
  - 頂点Bが頂点Aの子孫である場合
    →もとの閉路とは無関係の閉路ができる (C=Bとおく)
  - 頂点Bが頂点Aの子孫ではなく、頂点Aがもとの閉路上にある場合
    →もとの閉路の一部が変更されたような閉路ができる (C=Bとおく)
  - 頂点Bが頂点Aの子孫ではなく、頂点Aがもとの閉路上にない場合
    →もとの閉路が使われる

  - C=Bとおくことで、もとのグラフのdoublingを使って辿っても問題ない
    (一般には、もとのグラフ上を辿るのと変更後のグラフ上を辿るのでは異なる頂点に行き着く)

小課題1,2 (10+37点)

- 以上のアルゴリズムで$O(n^2 \log n)$が実現された

小課題3 (33点)

- 各頂点の行き先は互いに異なる

小課題3 (33点)

- 各頂点の行き先は互いに異なる
- →グラフは閉路が何個か集まったものになる

小課題3 (33点)

- 頂点1の連結成分の大きさをMとする。
- Aが頂点1とは異なる連結成分にあるとき → 一括処理
- Aが頂点1と同じ連結成分にあるとき
  - Bが頂点1とは異なる連結成分にあるとき
  - Bが頂点1と同じ連結成分にあるとき

小課題3 (33点)

- Aが頂点1と同じ連結成分にあり、Bが頂点1とは異なる連結成分にあるとき
  - この場合をまとめて処理すると、結果として頂点1とは異なる連結成分にある各頂点に$N - M$を足すのと同等であることがわかる。

小課題3 (33点)

- Aが頂点1と同じ連結成分にあり、Bが頂点1と同じ連結成分にあるとき
  - この結果できる閉路の大きさは1以上M以下。この大きさLごとに処理する
  - Lを固定すると、到達先としてありえる頂点はM/Lより少し多いくらいしかない
  - したがって処理の回数は
    $$\frac{M}{1}+\frac{M}{2}+\cdots+\frac{M}{M-1}+\frac{M}{M}$$
    程度となる。これは、$O(M\log M)$である。各処理を$O(1)$で行えばよい。

小課題3 (33点)

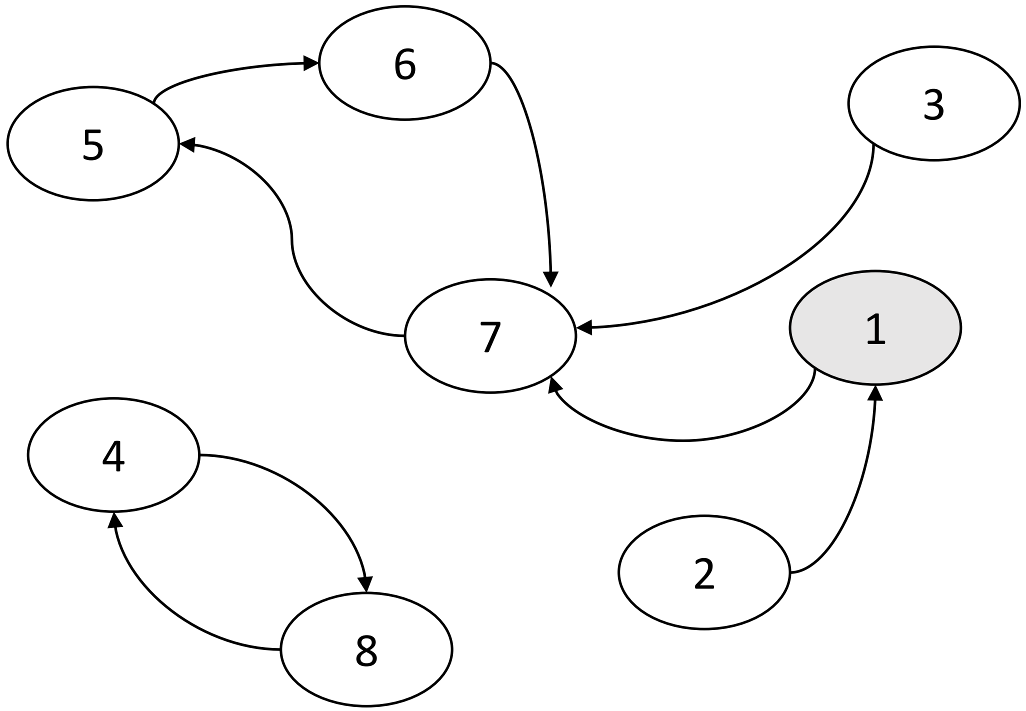
- 以上により、$O(n \log n)$で処理できた

満点解法

- グラフを例示しつつ説明

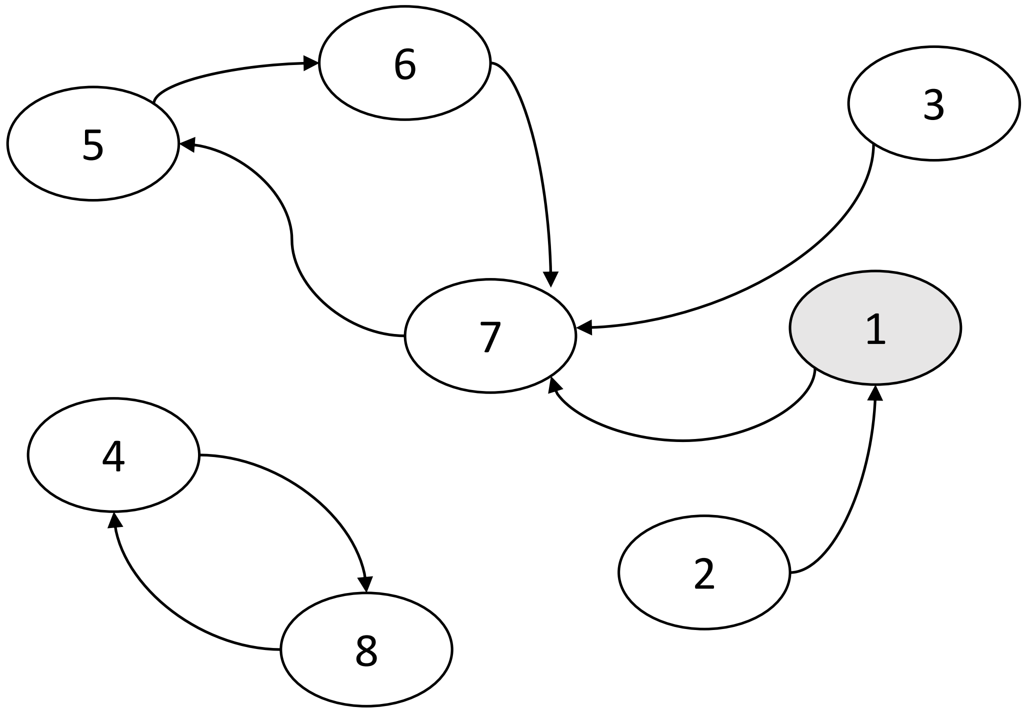
満点解法

- このグラフはN=8, K=10



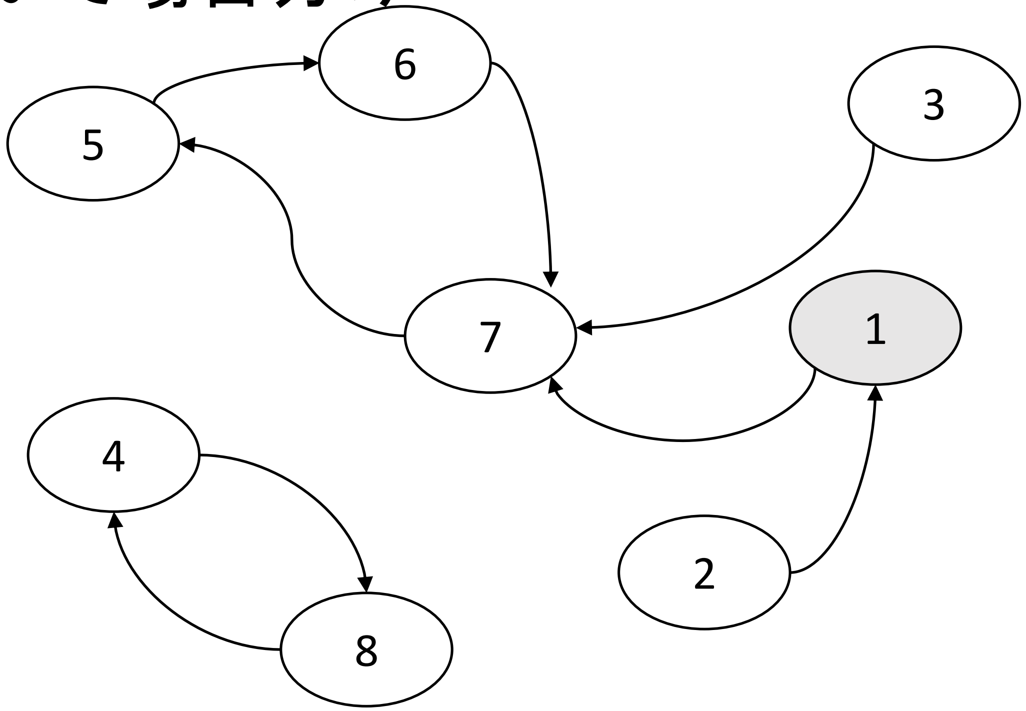
満点解法

- 行き先が変更される頂点をA, その頂点の新しい行き先をBとする



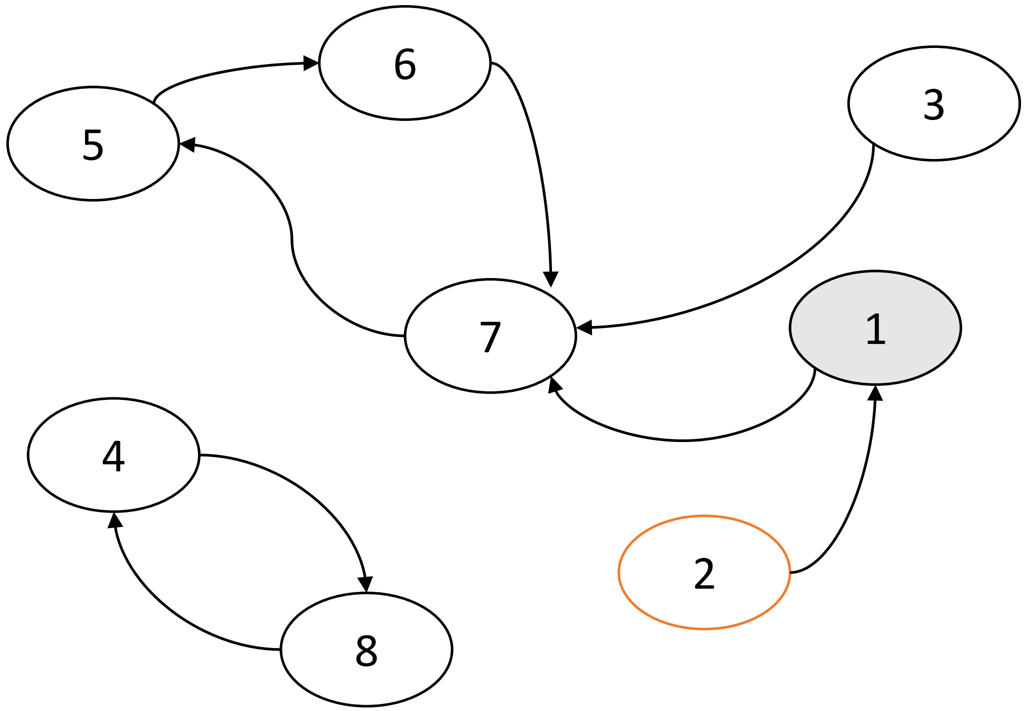
満点解法

- 行き先が変更される頂点をA, その頂点の新しい行き先をBとする
- そもそも頂点1からAに到達できるかで場合分け



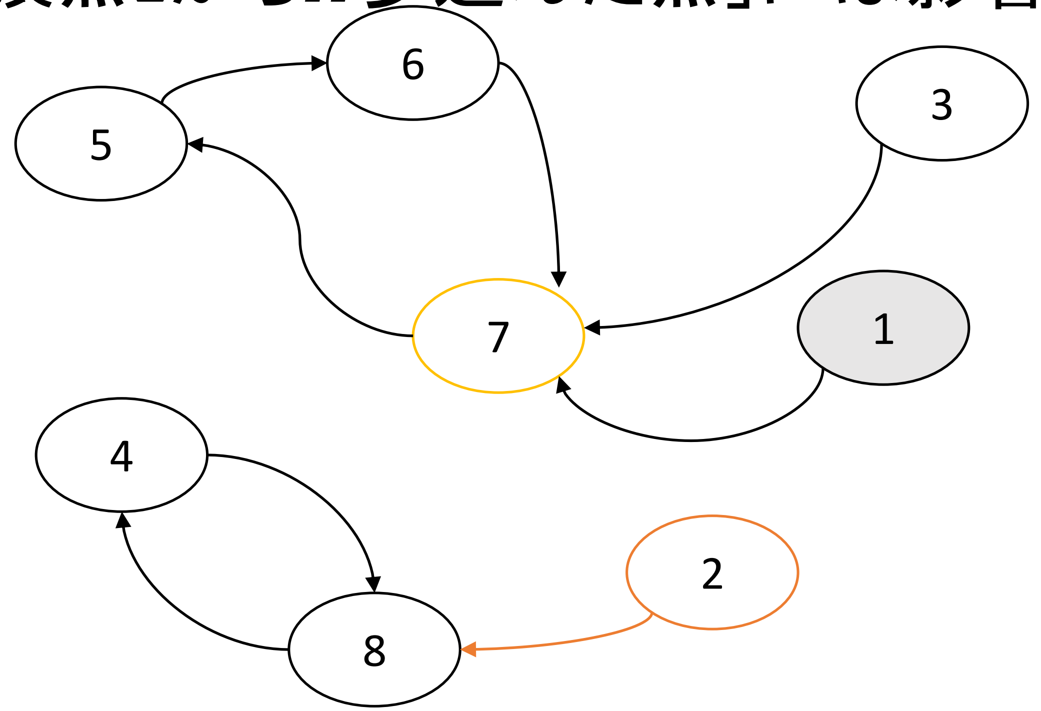
満点解法 – 場合分け(1)

- 頂点1からAに到達できない場合



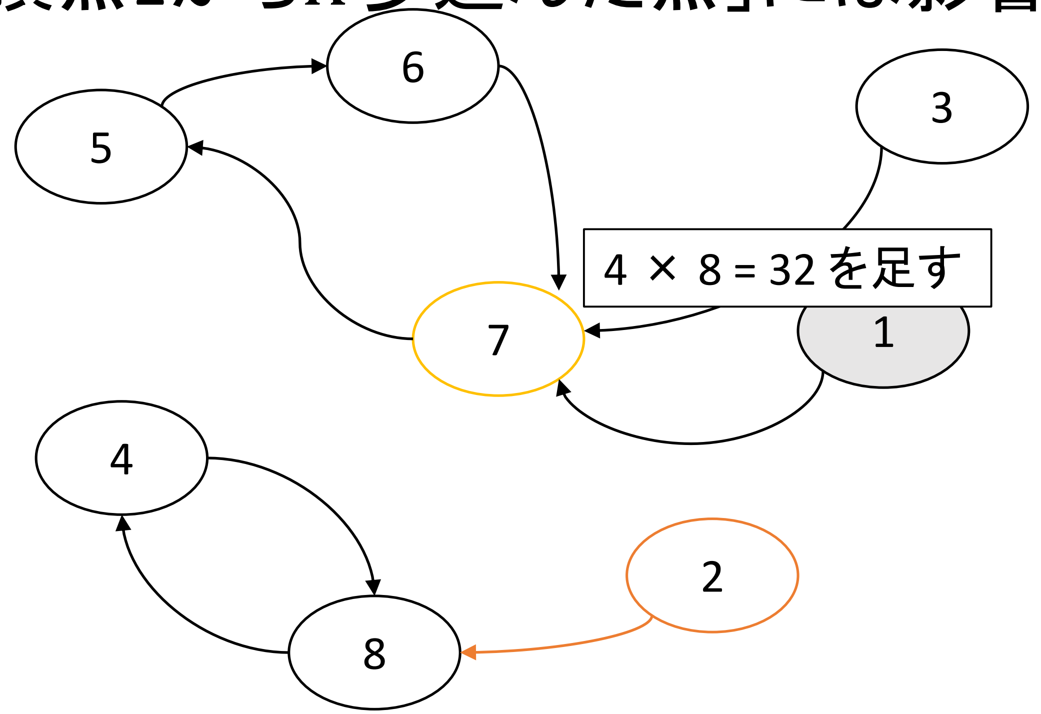
満点解法 – 場合分け(1)

- 頂点1からAに到達できない場合
- このように行き先が変わっても、「頂点1からK歩進んだ点」には影響しない



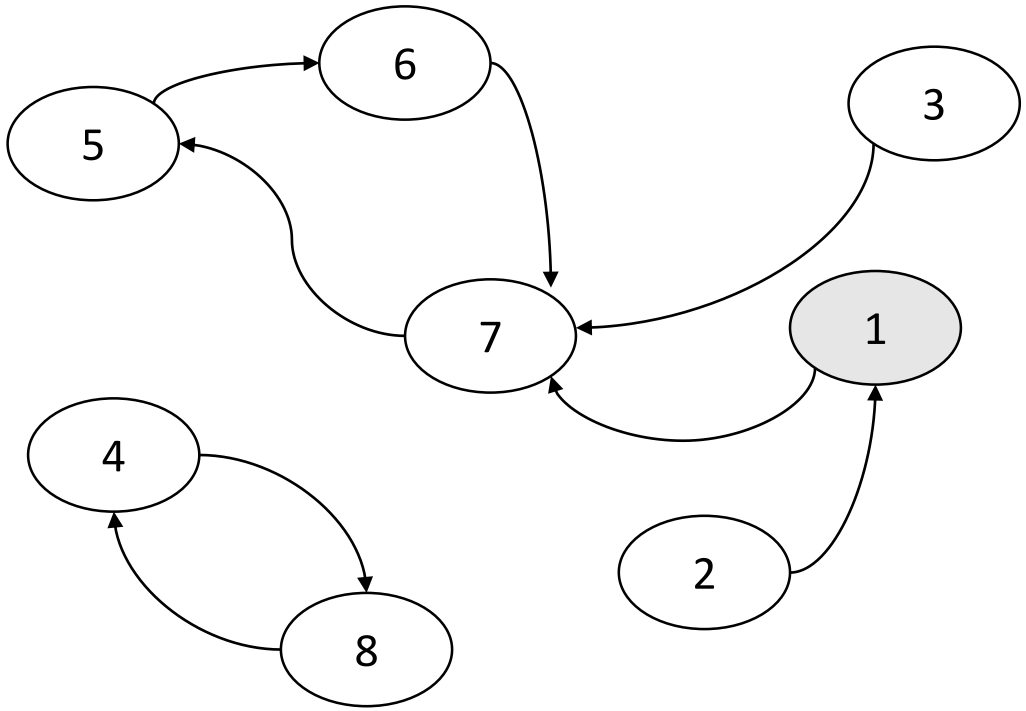
満点解法 – 場合分け(1)

- 頂点1からAに到達できない場合
- このように行き先が変わっても、「頂点1からK歩進んだ点」には影響しない
- (頂点1から到達不能な点の個数)$\times N$
  を1からK歩進んだ先の点に足す



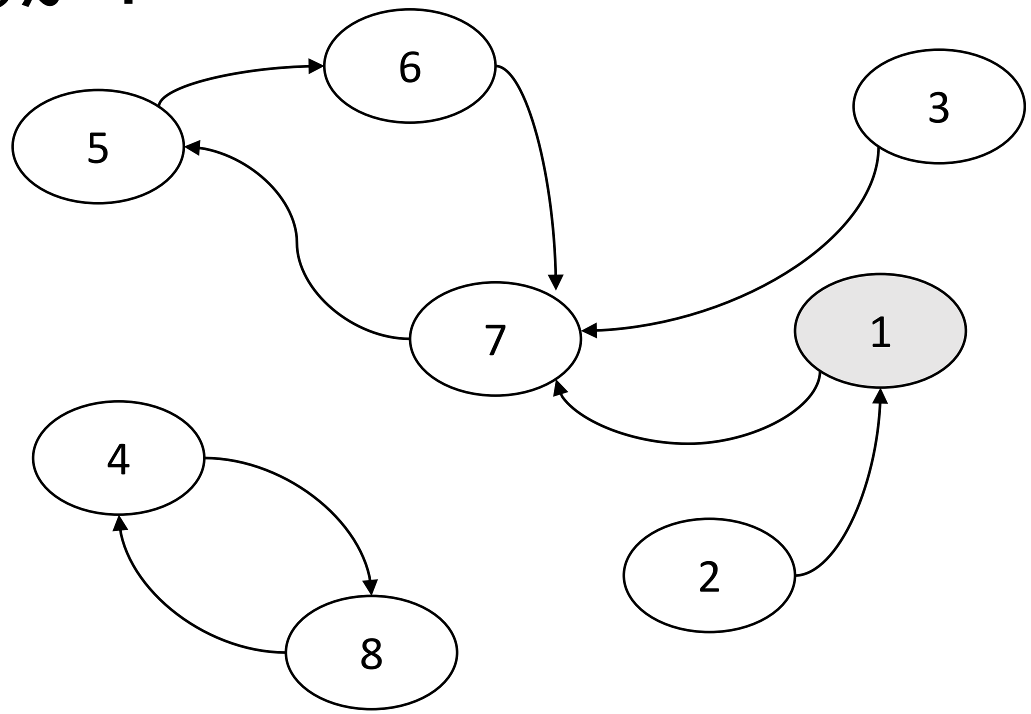
満点解法

- 以下、Aが頂点1から到達可能の場合を調べる



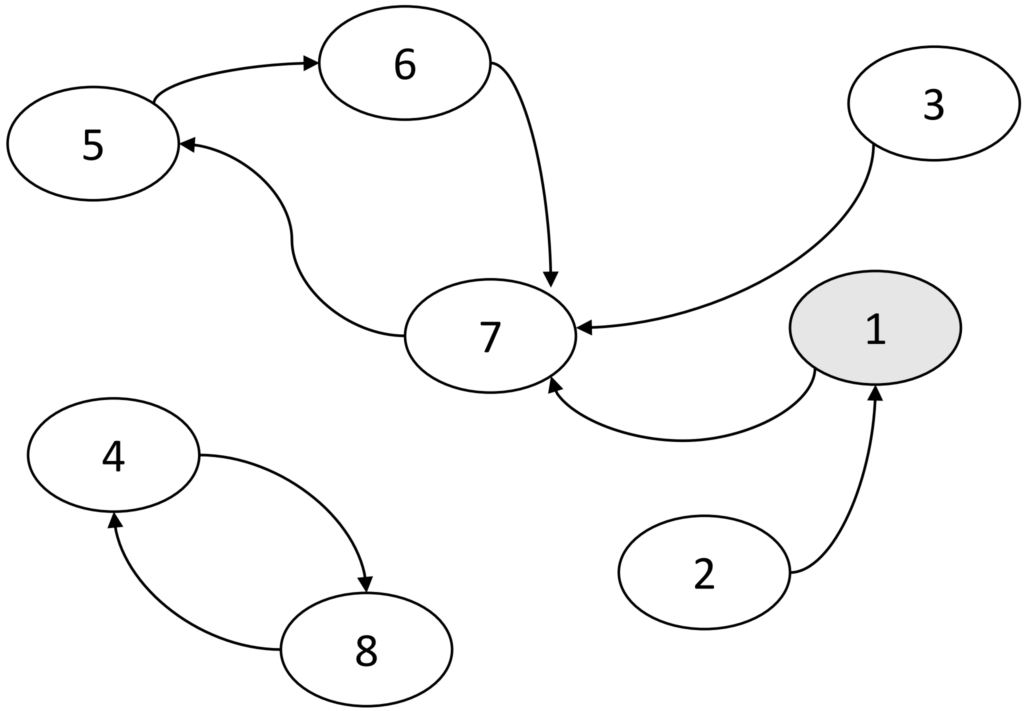
満点解法

- 以下、Aが頂点1から到達可能の場合を調べる
- Bは頂点1と同じ連結成分に属するか？



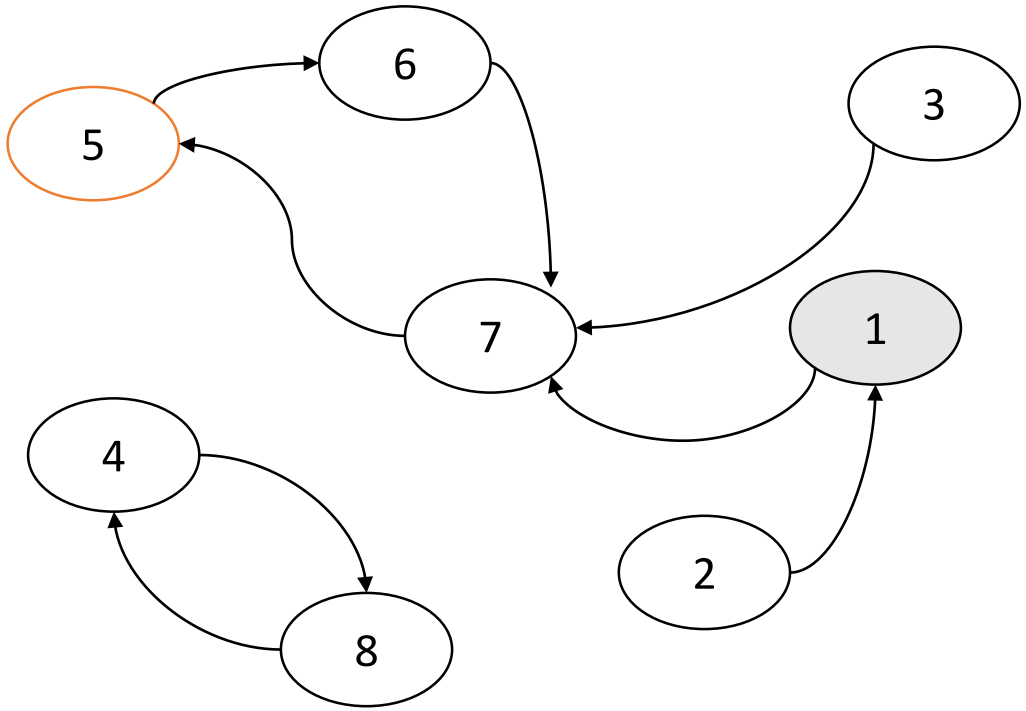
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合



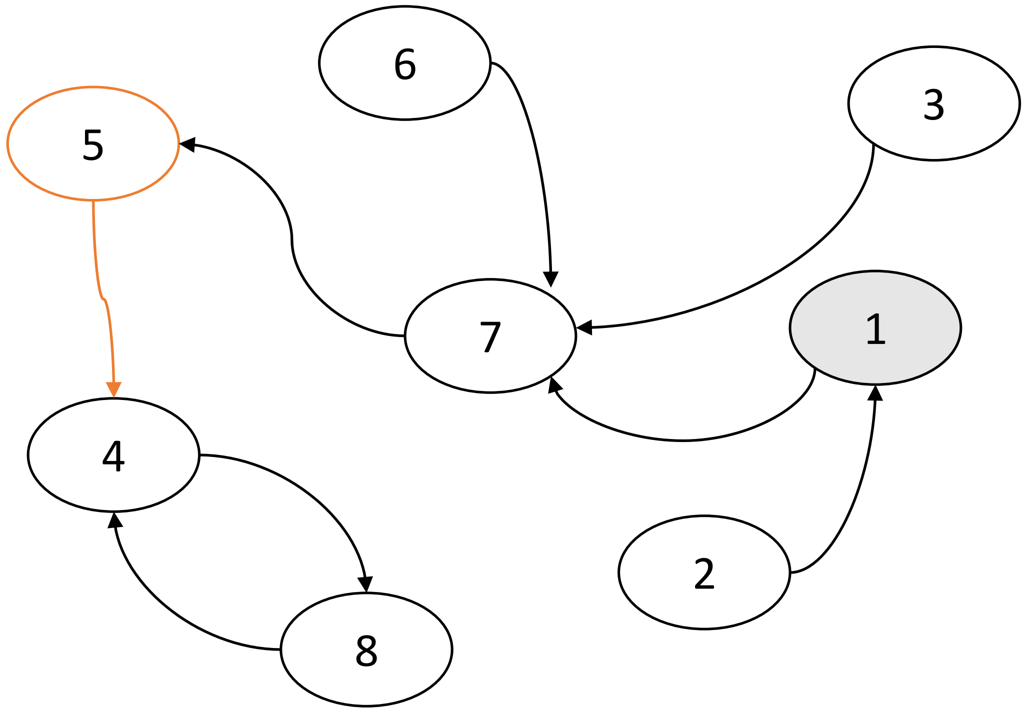
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合



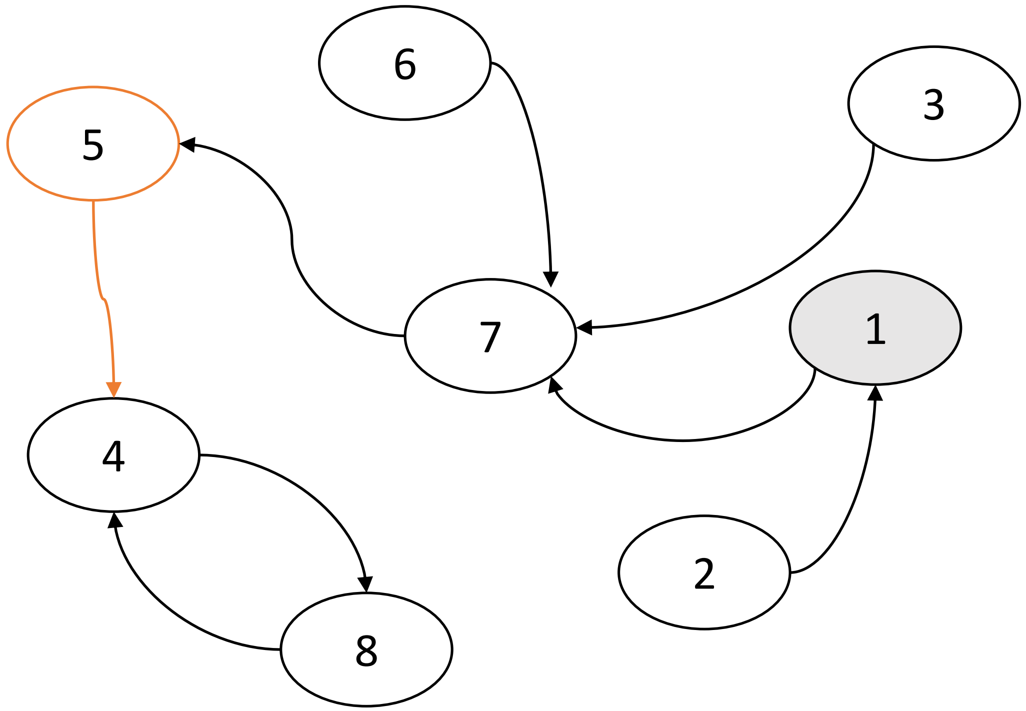
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合



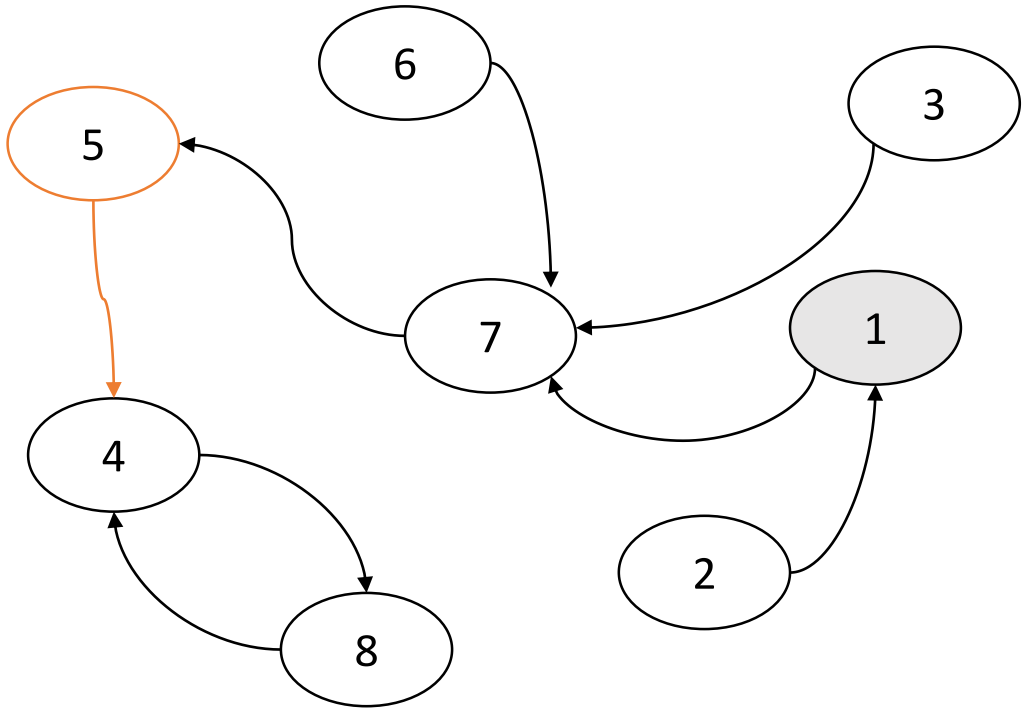
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える



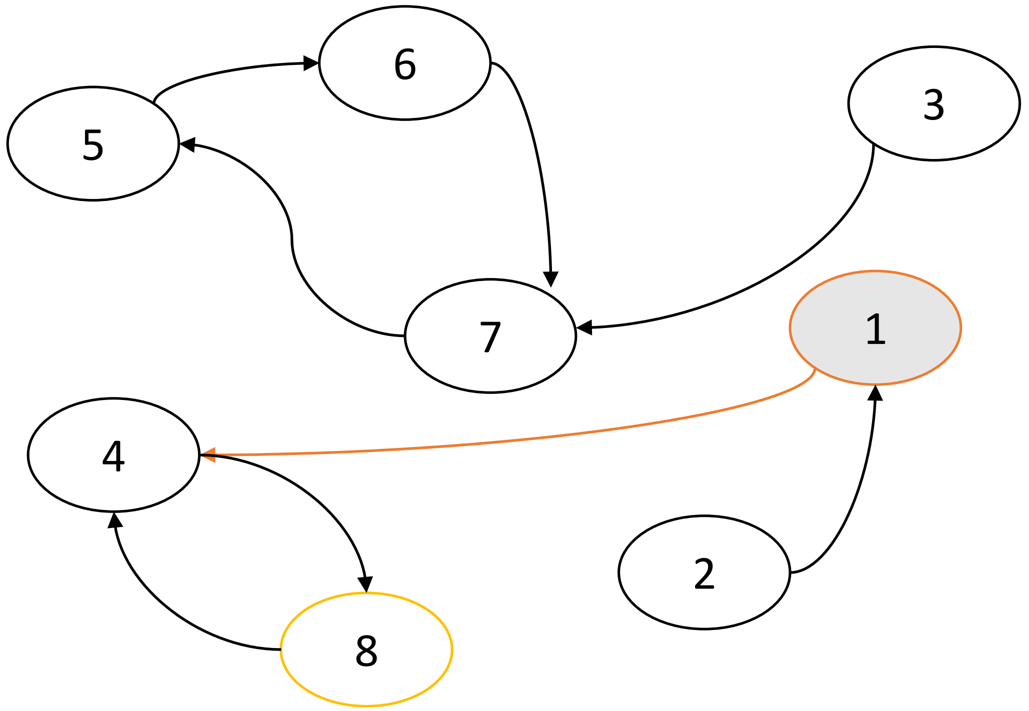
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定



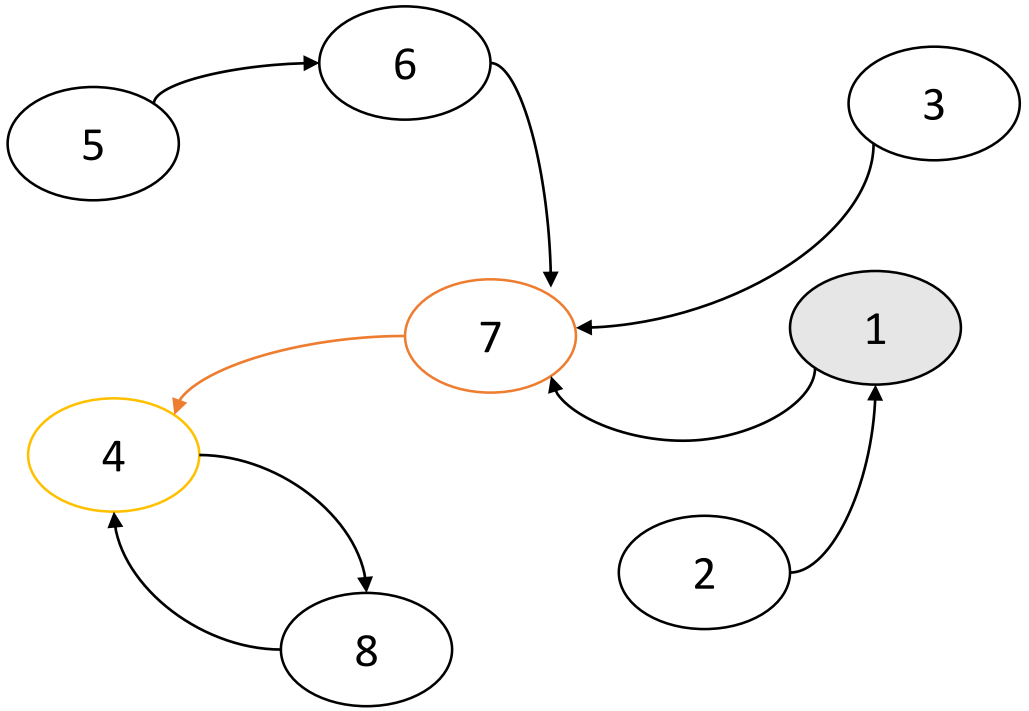
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、到達先は隣に移動していく



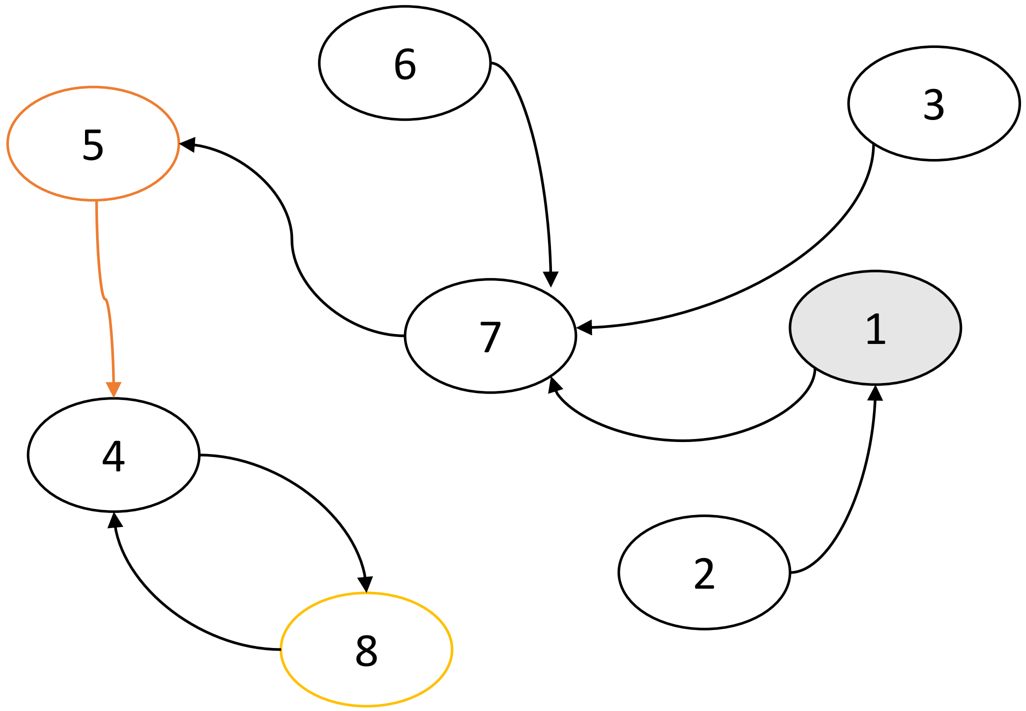
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、到達先は隣に移動していく



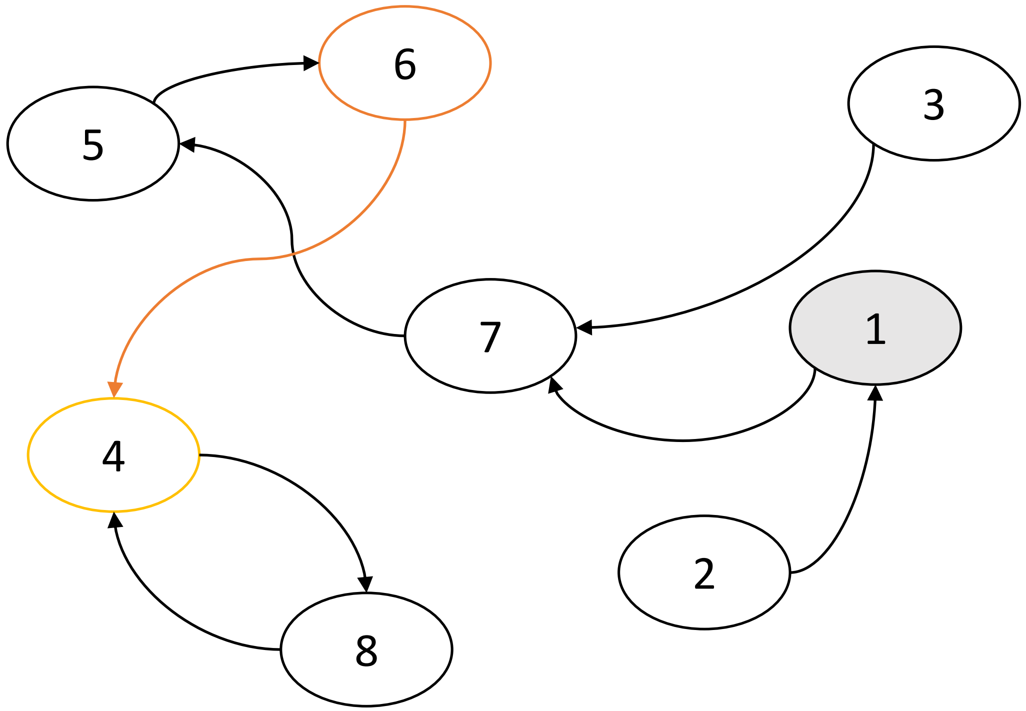
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、到達先は隣に移動していく



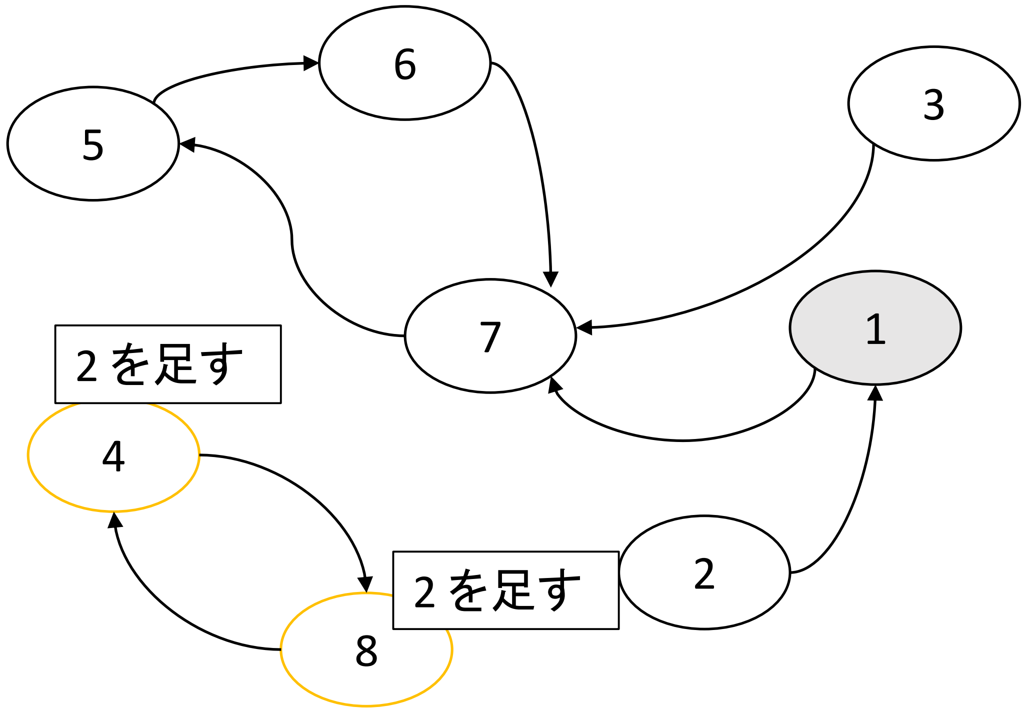
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、
  到達先は隣に移動していく



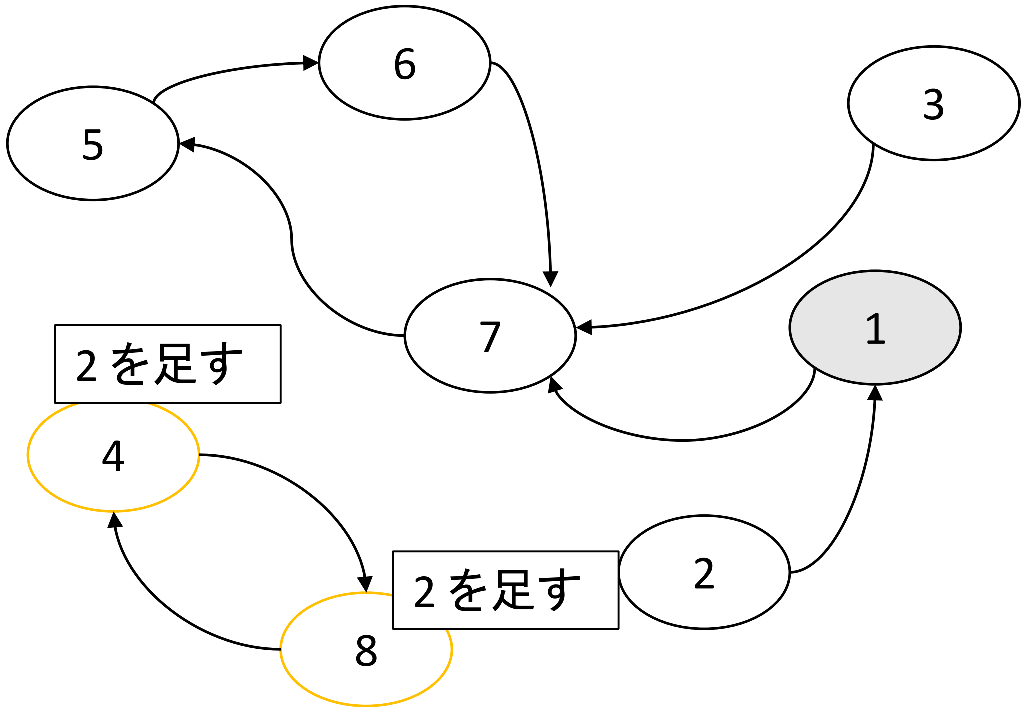
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、到達先は隣に移動していく
- 連続した区間に1を足す→所謂imos法でできる(ループに注意)



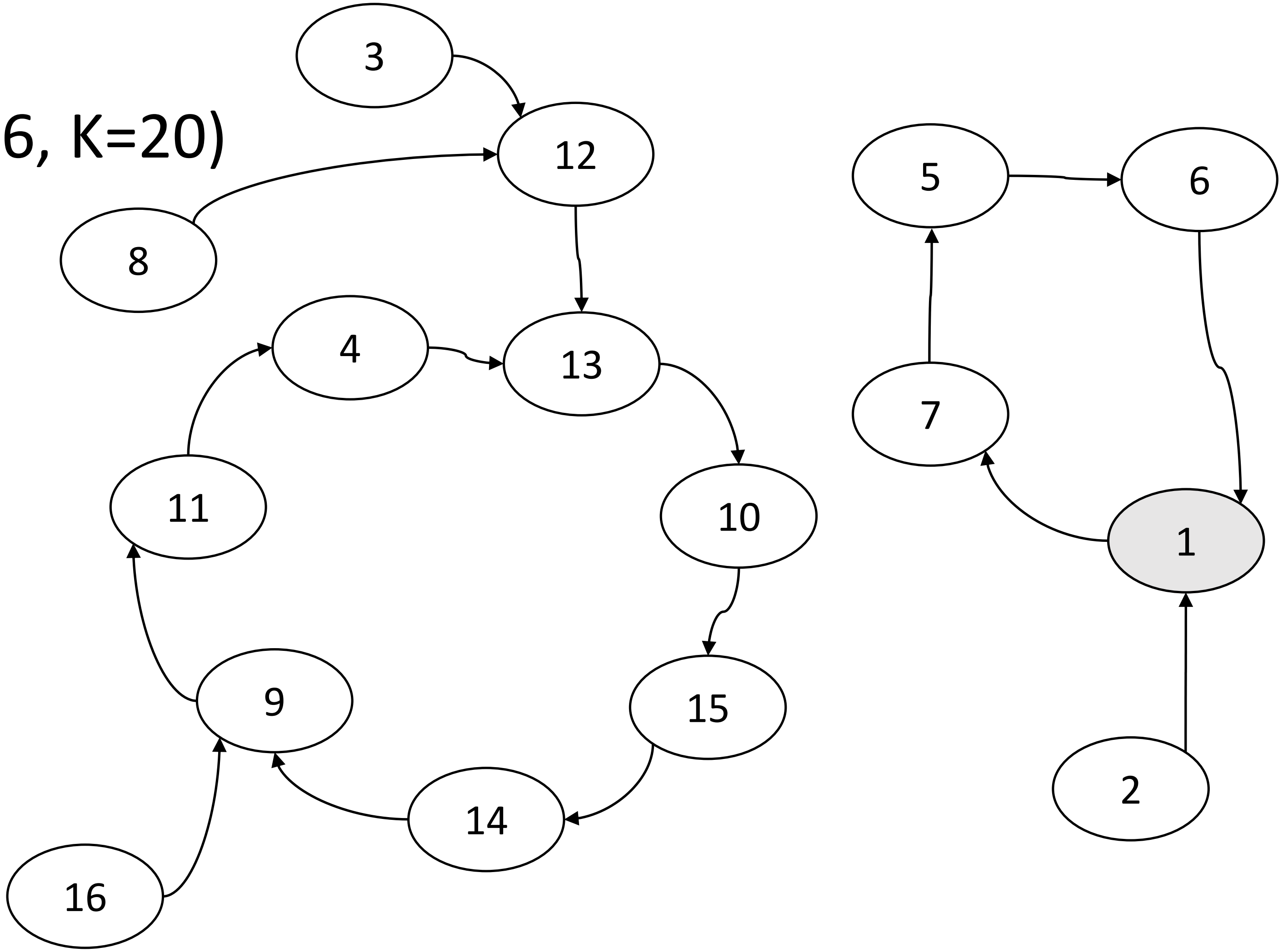
満点解法 – 場合分け(2)

- Bが頂点1とは異なる連結成分に属する場合
- Bを固定して考える
- →ループの長さは一定
- Aを頂点1から順番に動かすと、到達先は隣に移動していく
- 連続した区間に1を足す→所謂imos法でできる
(ループに注意。
この場合2周している)



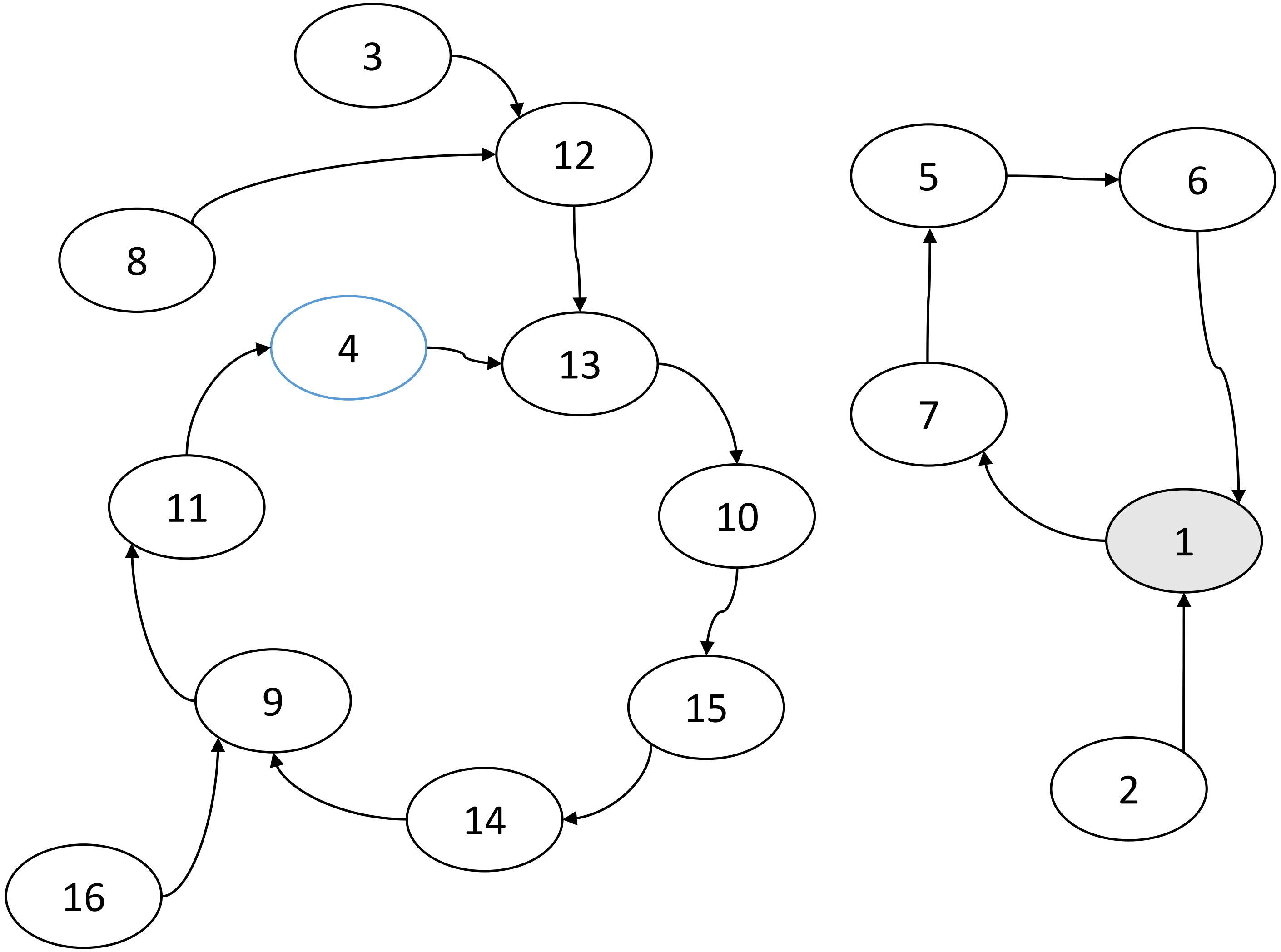
満点解法 – 場合分け(2)

- 別のグラフで再説明(N=16, K=20)



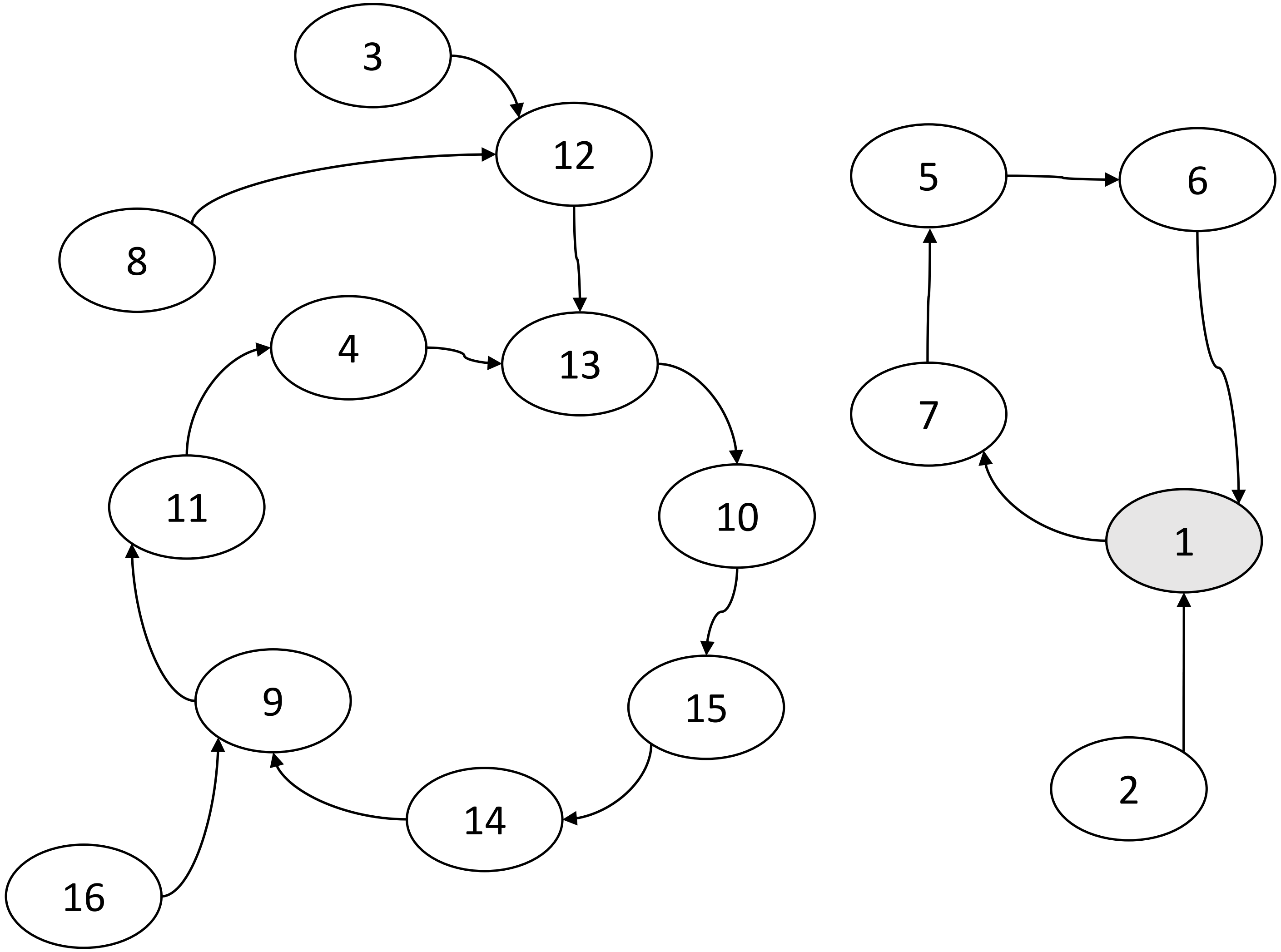
満点解法 – 場合分け(2)

- 頂点4をこの連結成分の「根」としておく



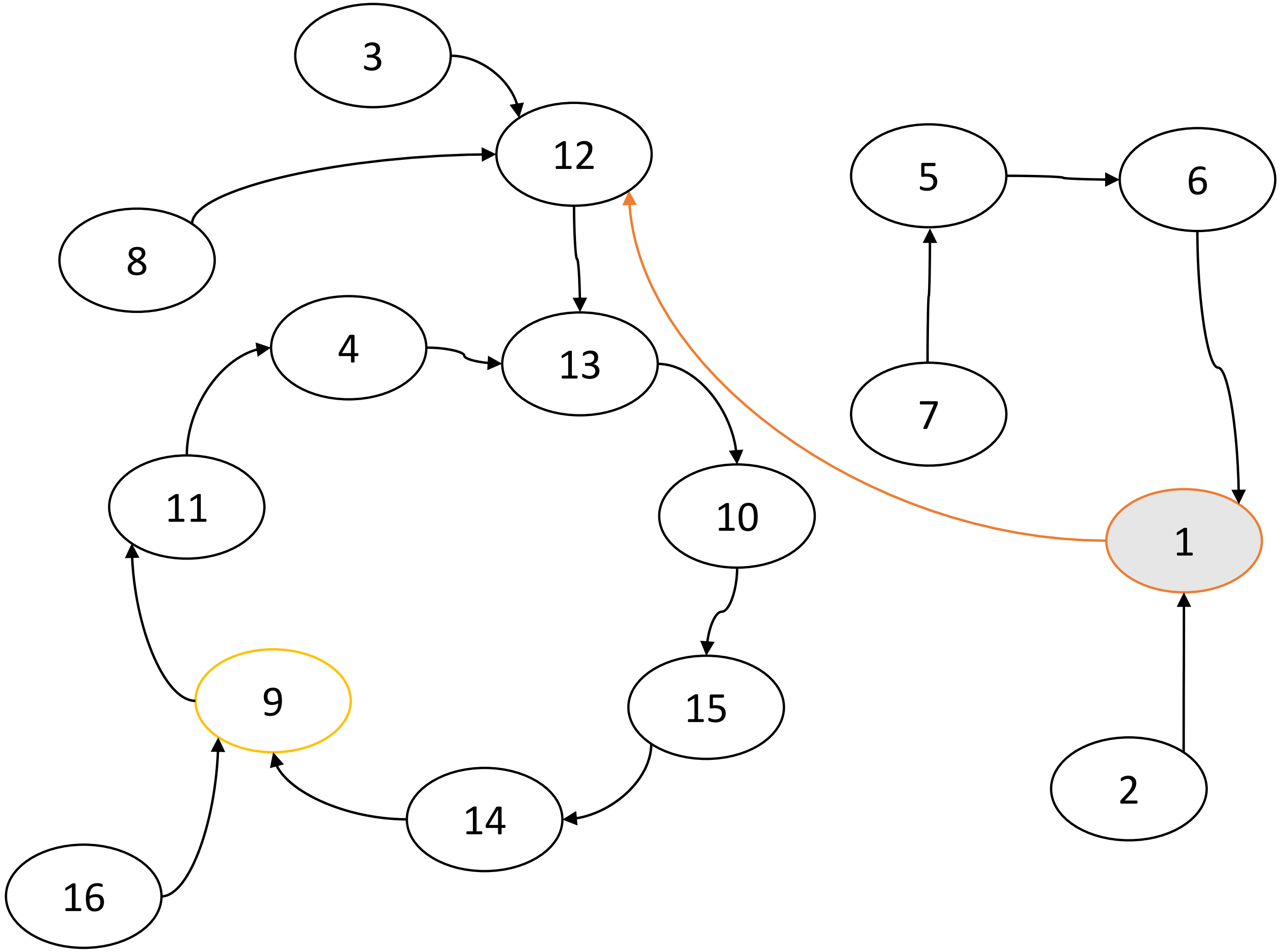
満点解法 – 場合分け(2)

- B=12の場合を考える



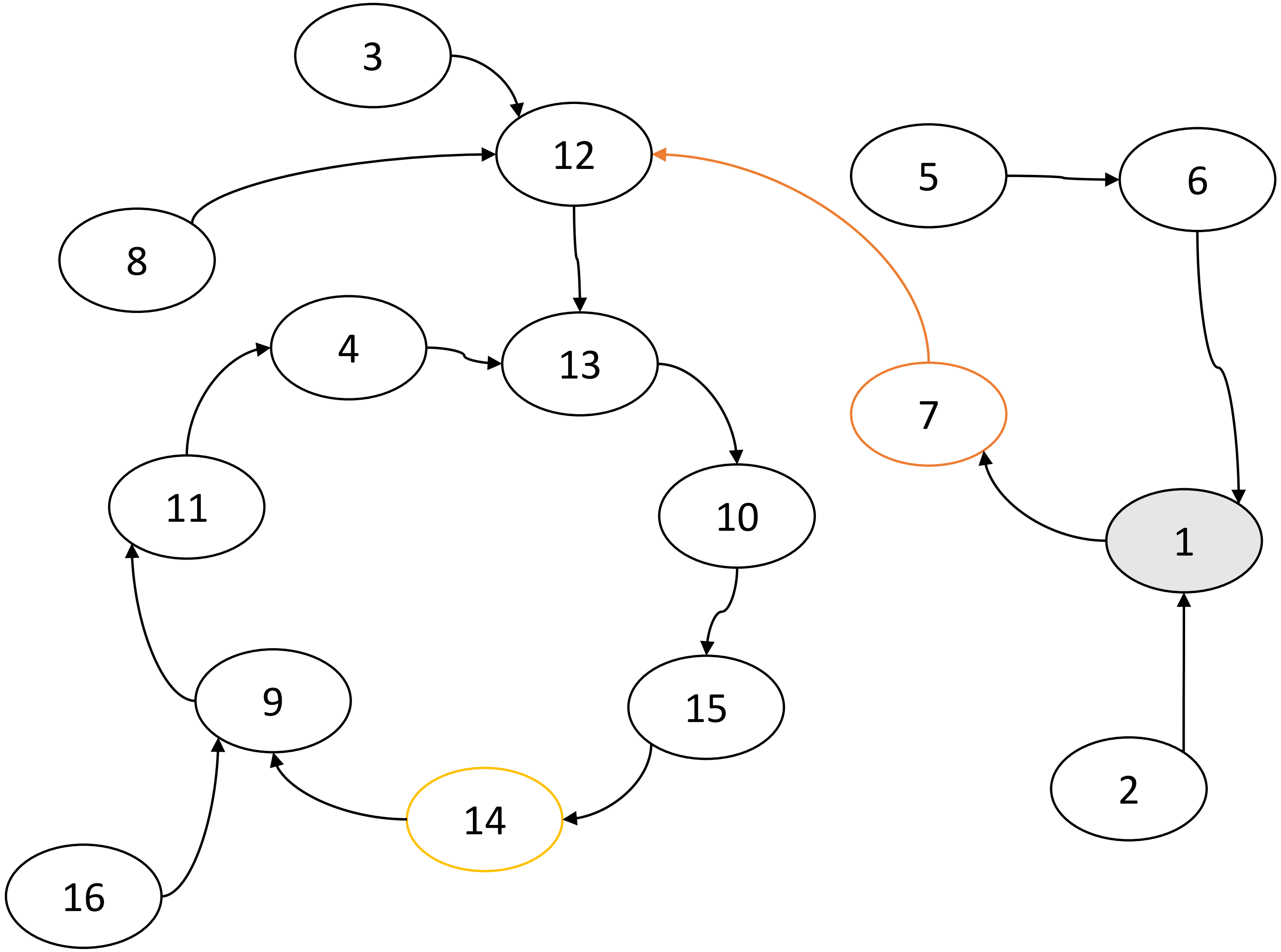
満点解法 – 場合分け(2)

- B=12の場合を考える



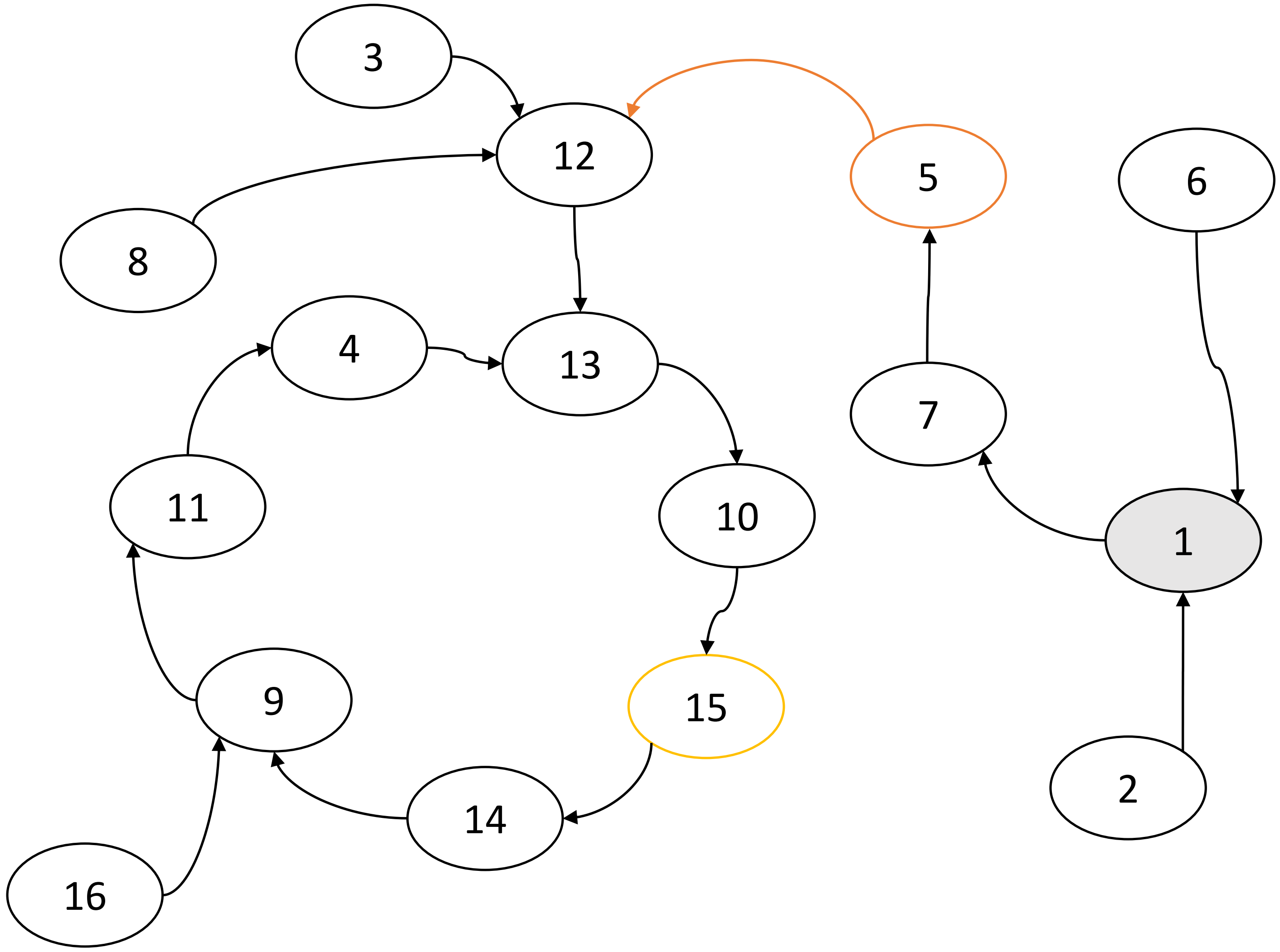
満点解法 – 場合分け(2)

- B=12の場合を考える



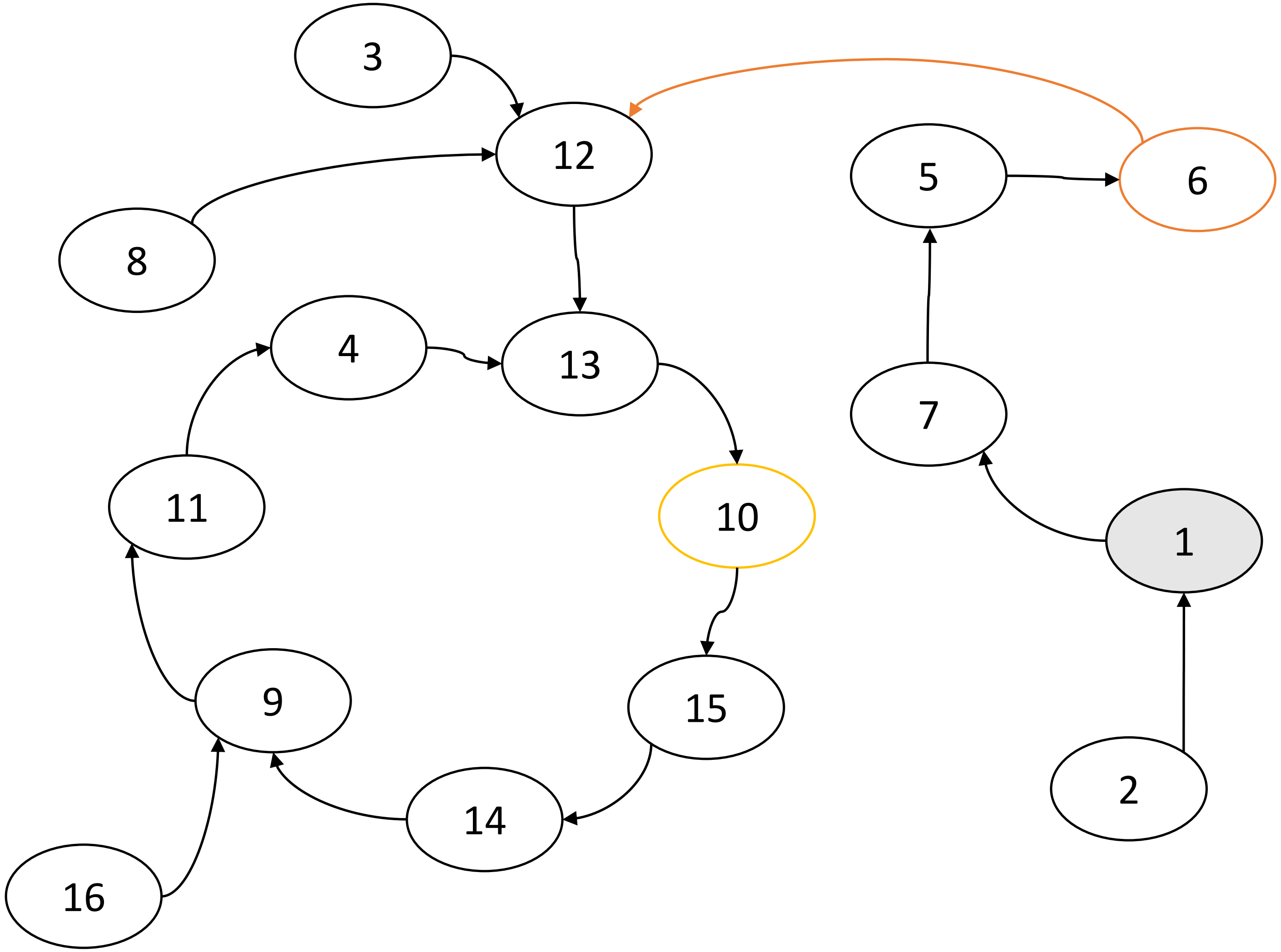
満点解法 – 場合分け(2)

- B=12の場合を考える



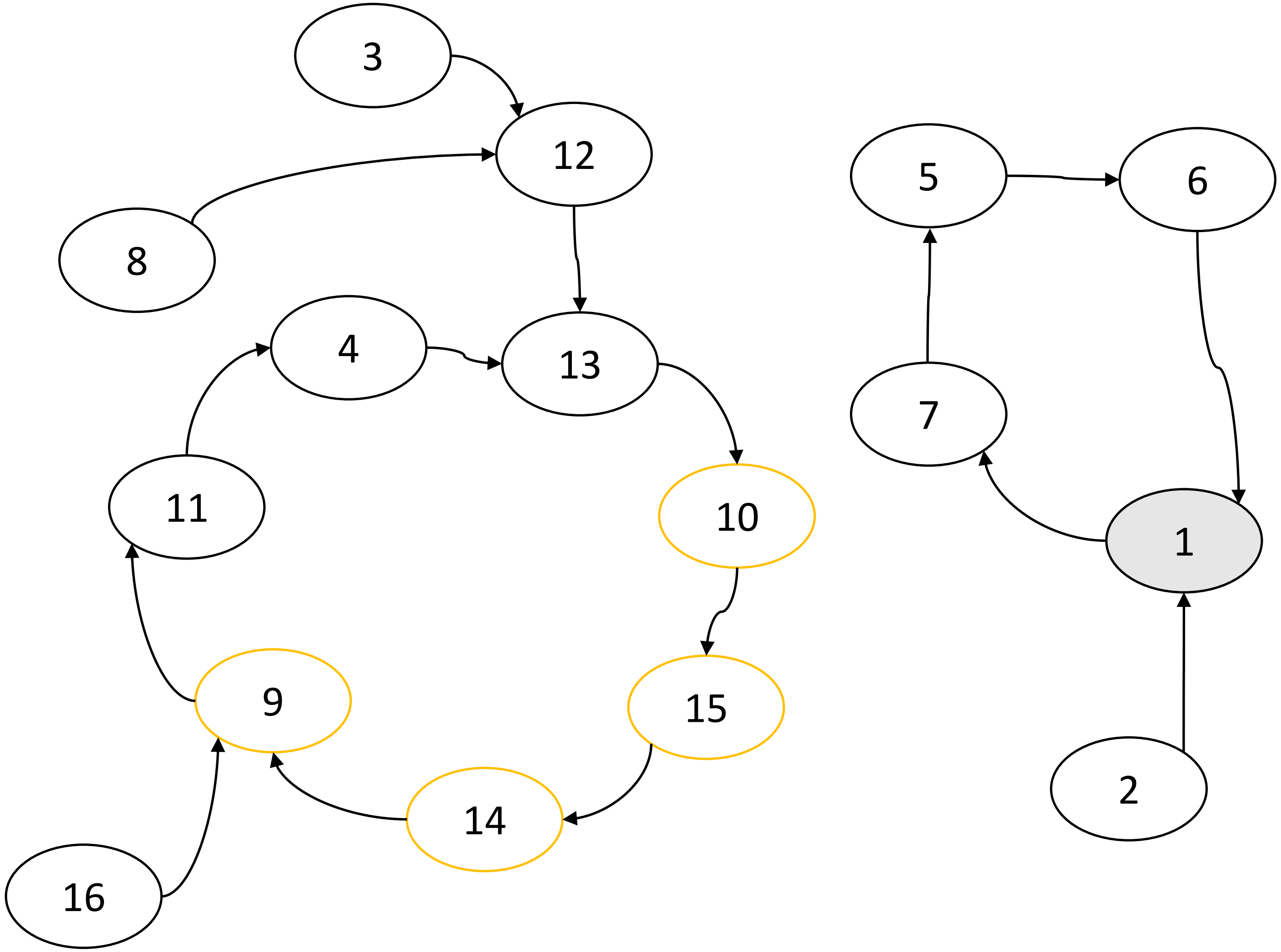
満点解法 – 場合分け(2)

- B=12の場合を考える



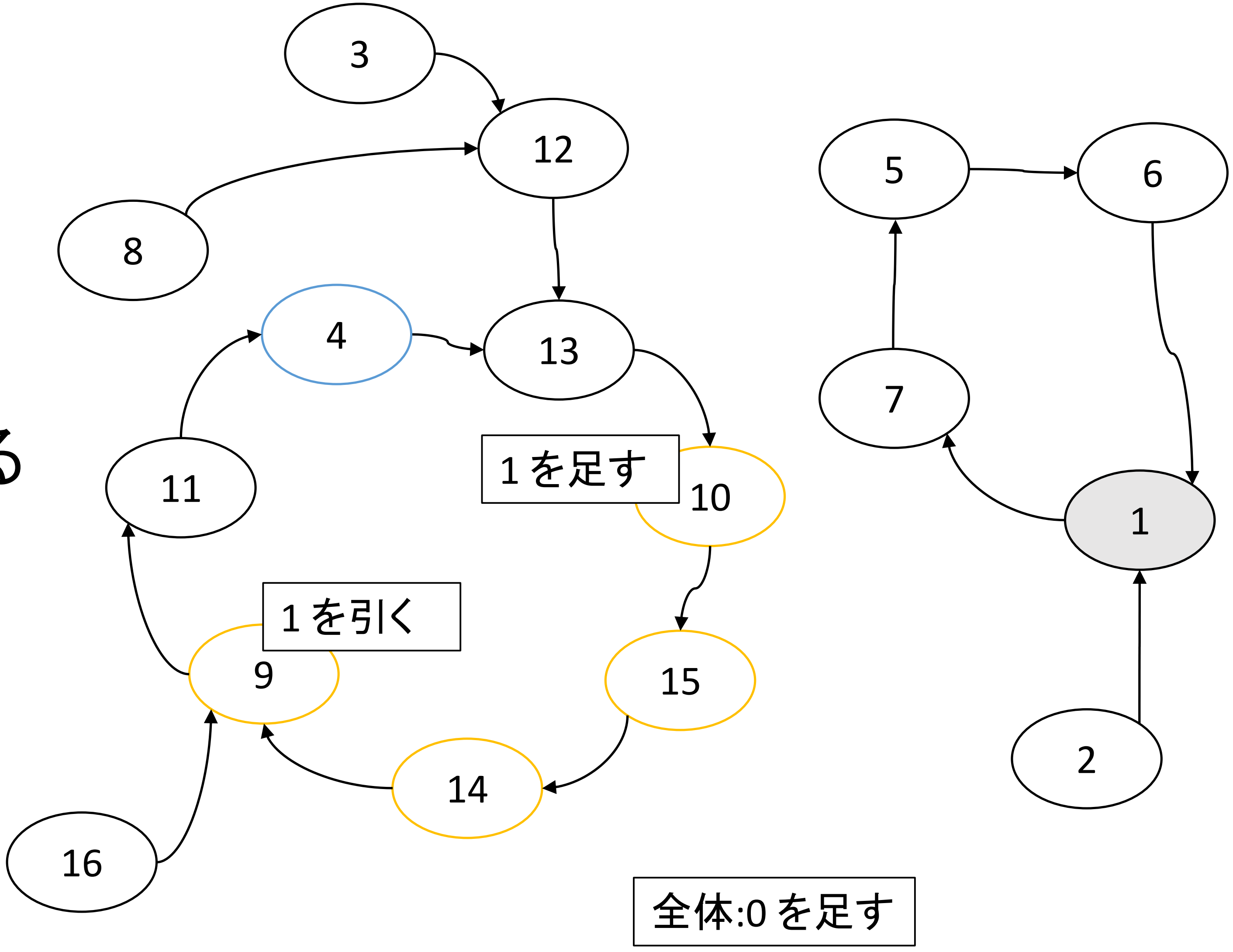
満点解法 – 場合分け(2)

- B=12の場合を考える
- 頂点10,15,14,9に1ずつ足せばよいことがわかった



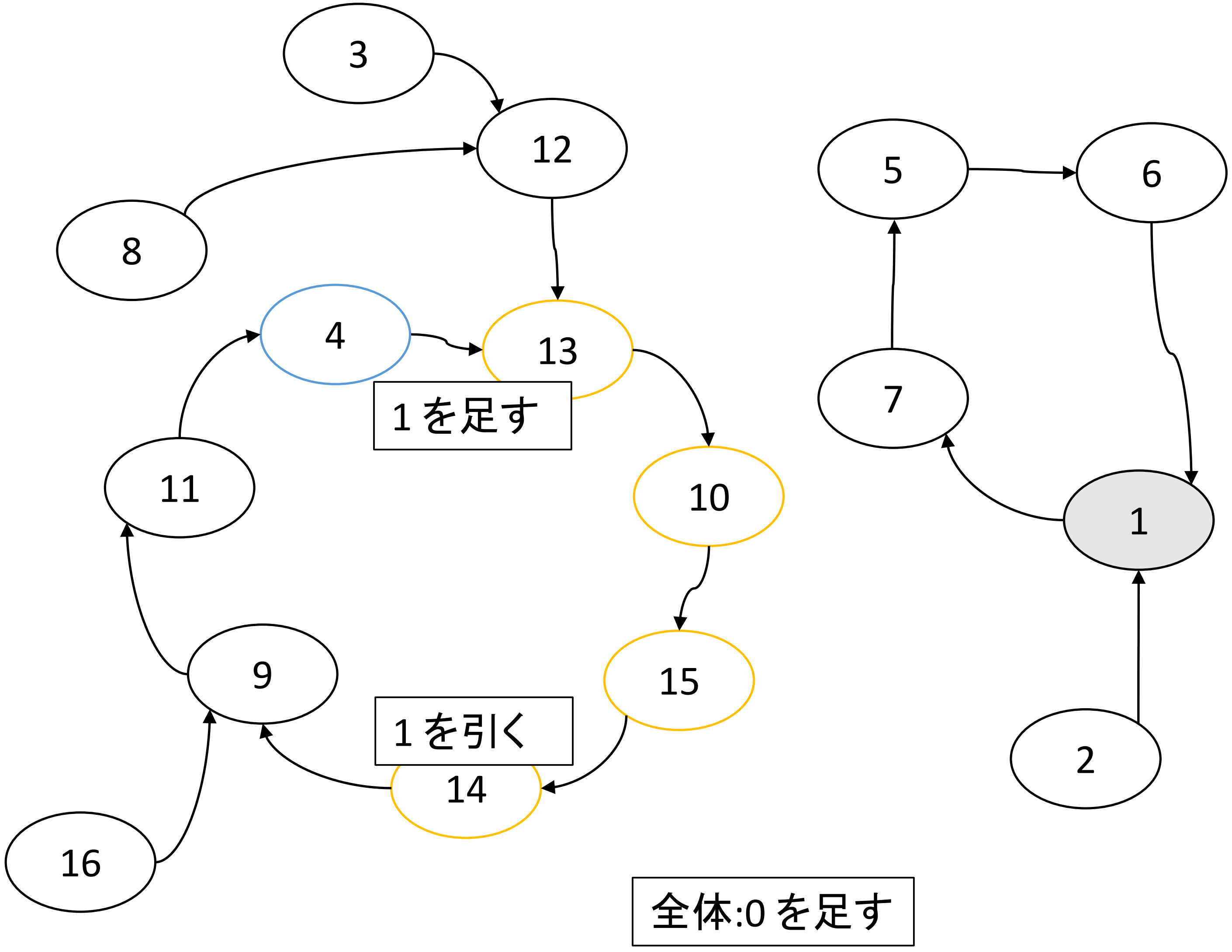
満点解法 – 場合分け(2)

- B=12の場合を考える
- 頂点10,15,14,9に1ずつ足せばよいことがわかった
- これをこのように記録する



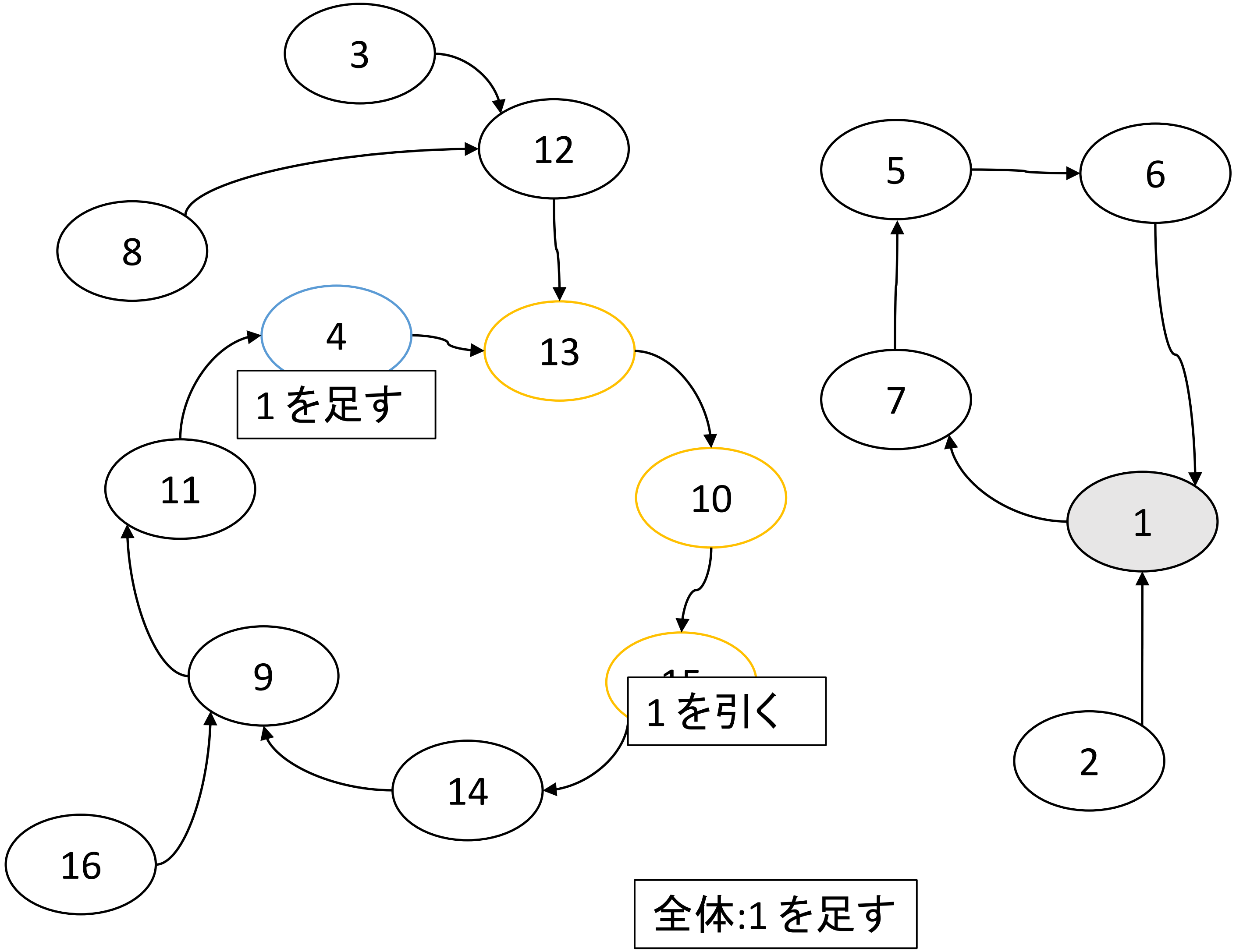
満点解法 – 場合分け(2)

- B=3,8,11の場合



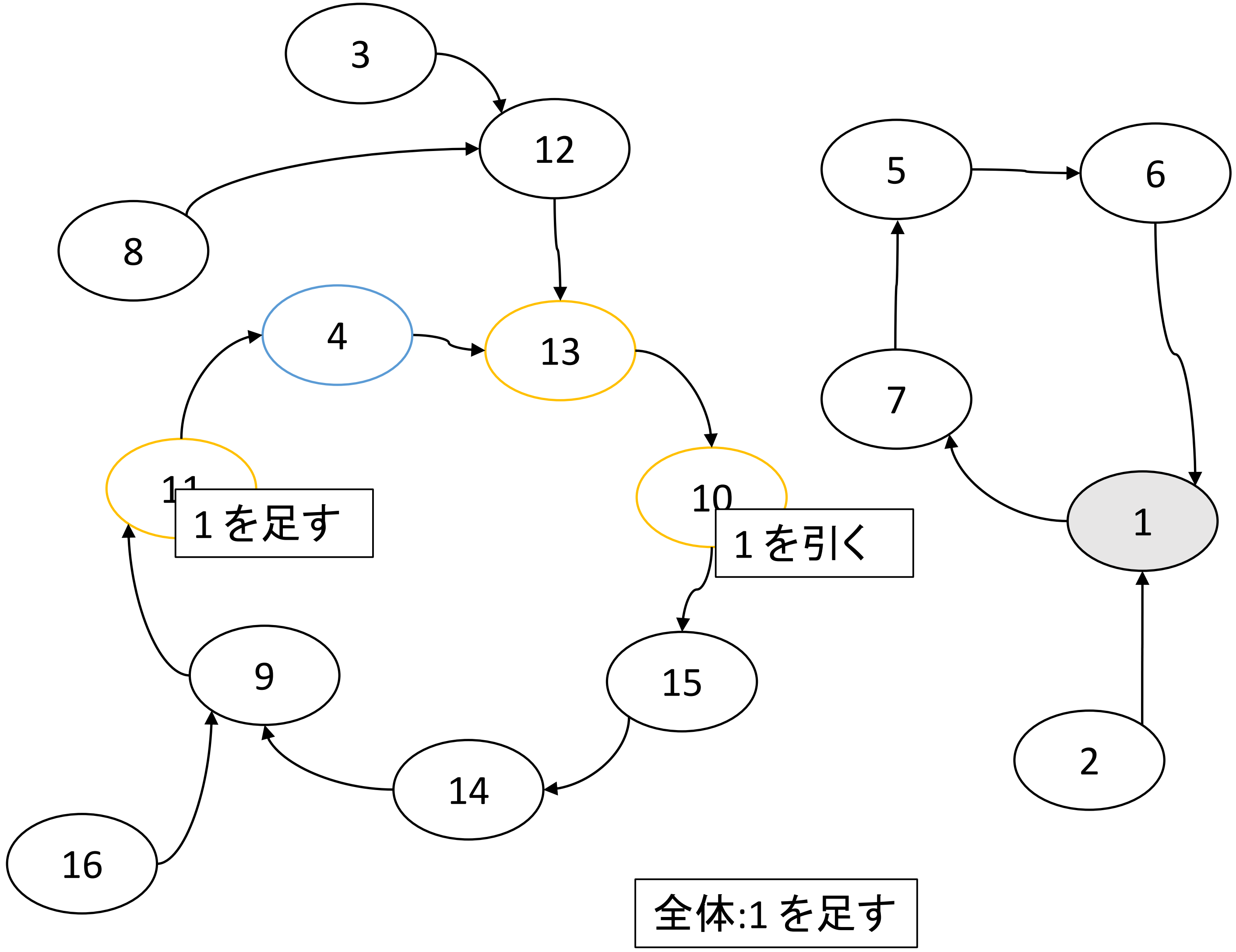
満点解法 – 場合分け(2)

- B=9の場合



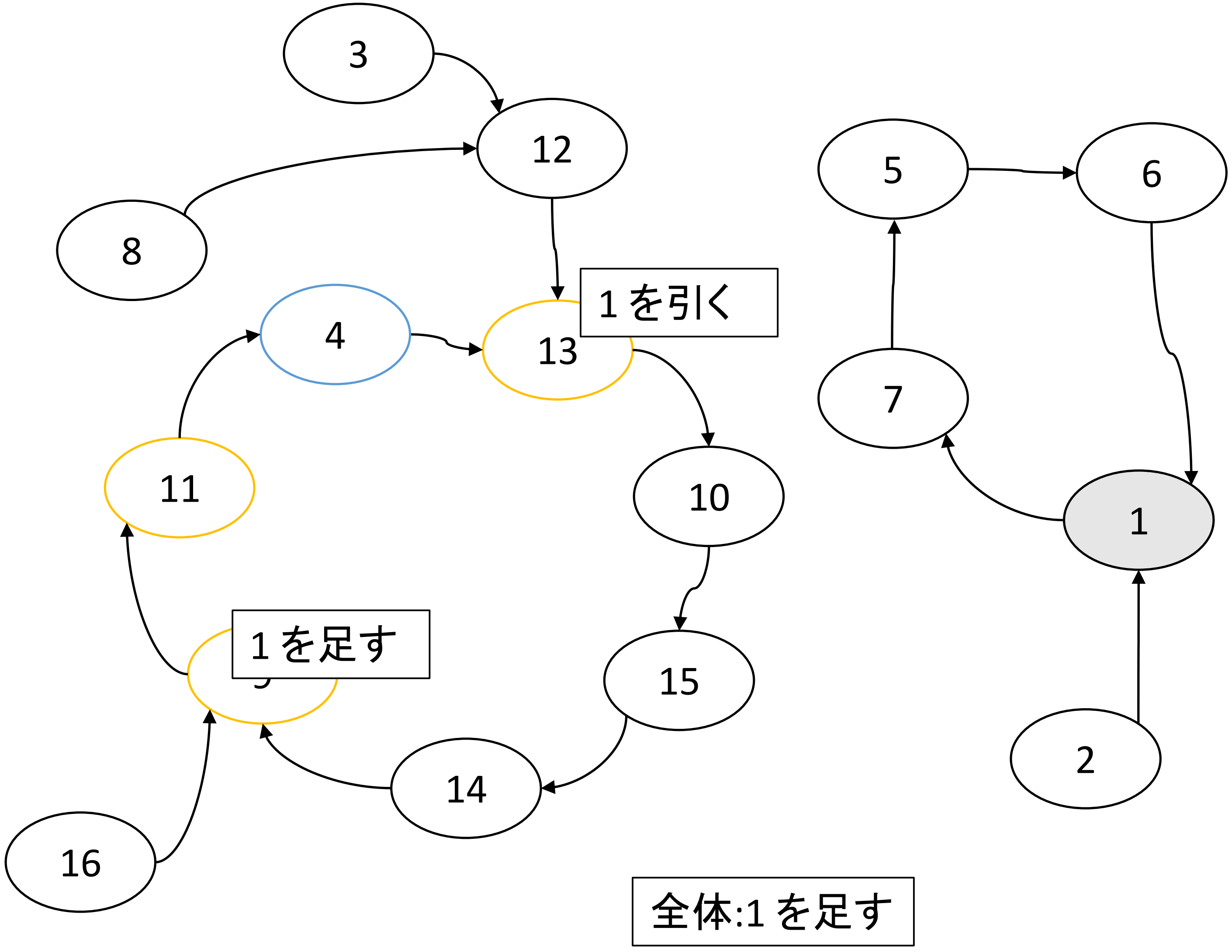
満点解法 – 場合分け(2)

- B=14,16の場合



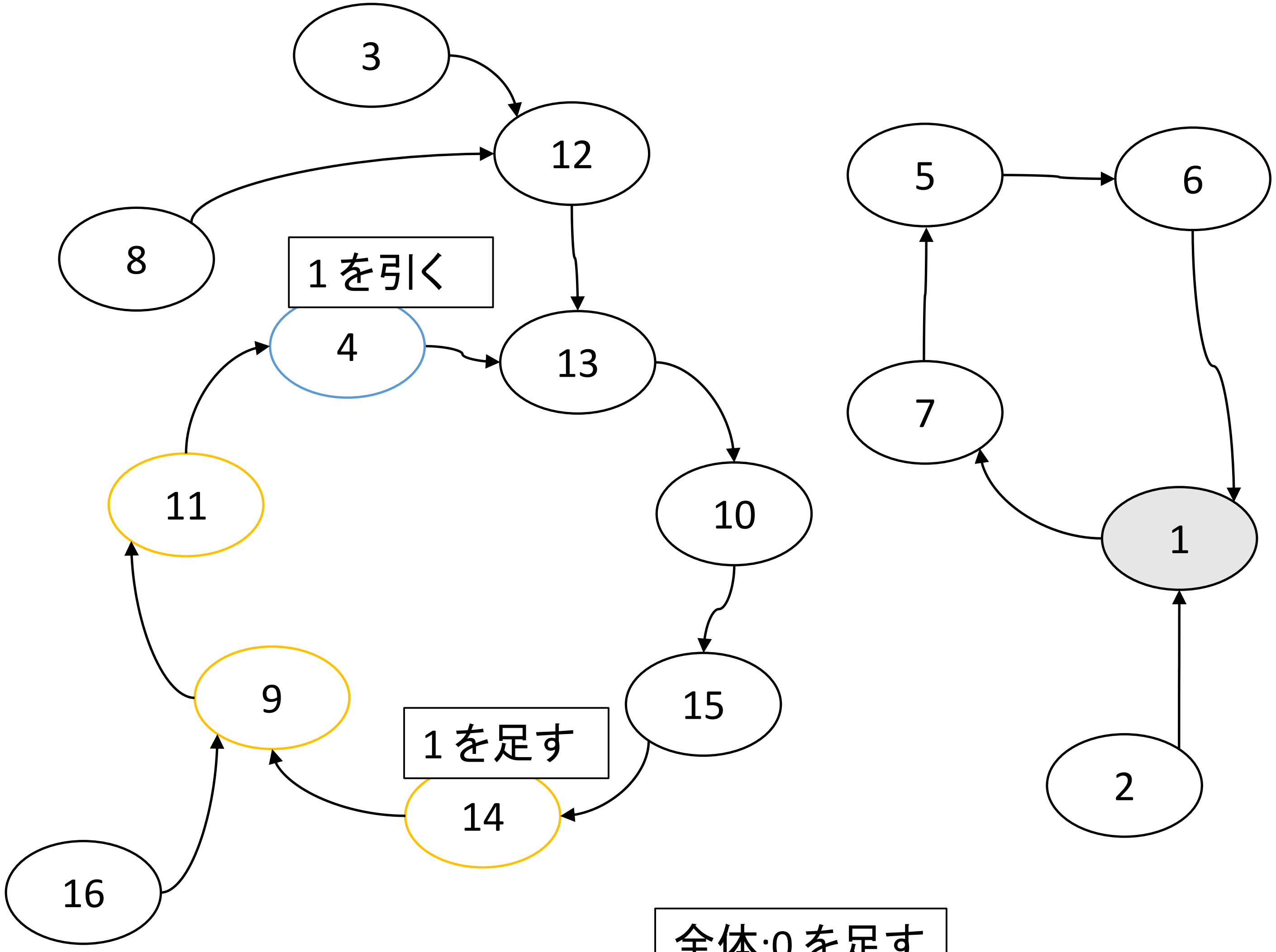
満点解法 – 場合分け(2)

- B=15の場合



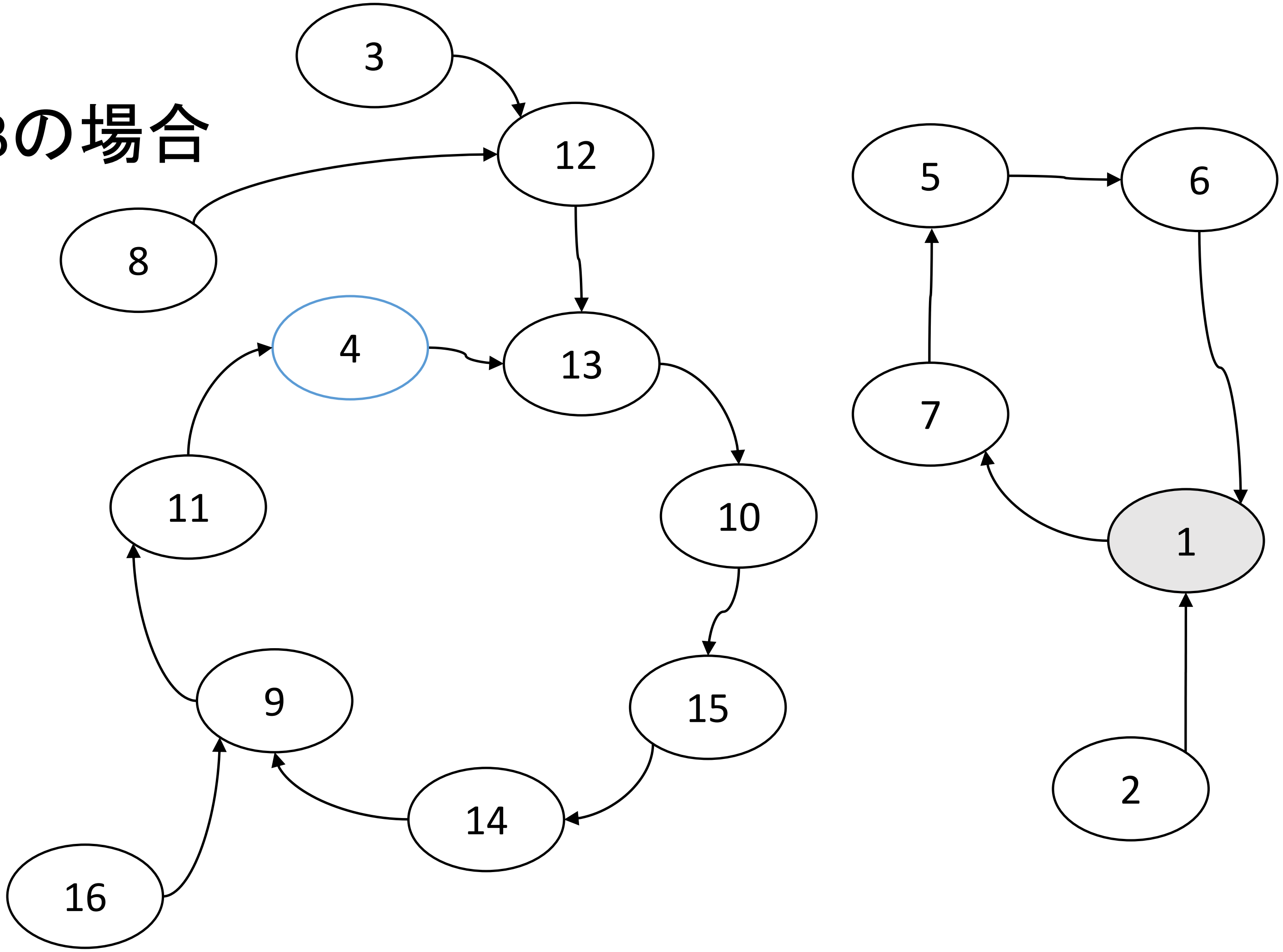
満点解法 – 場合分け(2)

- B=10の場合



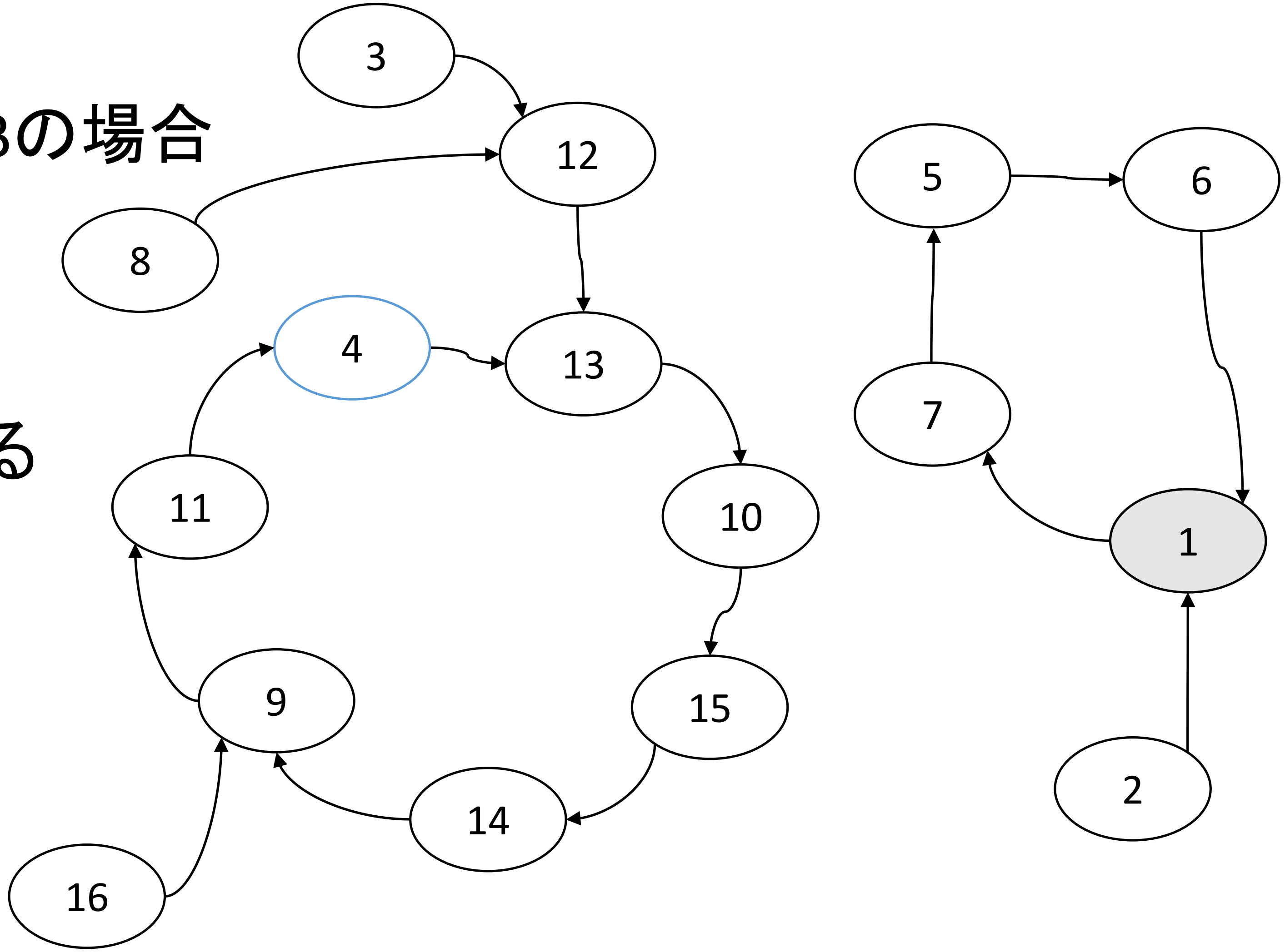
満点解法 – 場合分け(2)

- 連結成分内の各頂点がBの場合を全て処理する



満点解法 – 場合分け(2)

- 連結成分内の各頂点がBの場合を全て処理する
- →頂点13から頂点4に向かって部分和をとると本来のカウントが得られる
- (ループ全体に対するカウントを忘れずに)

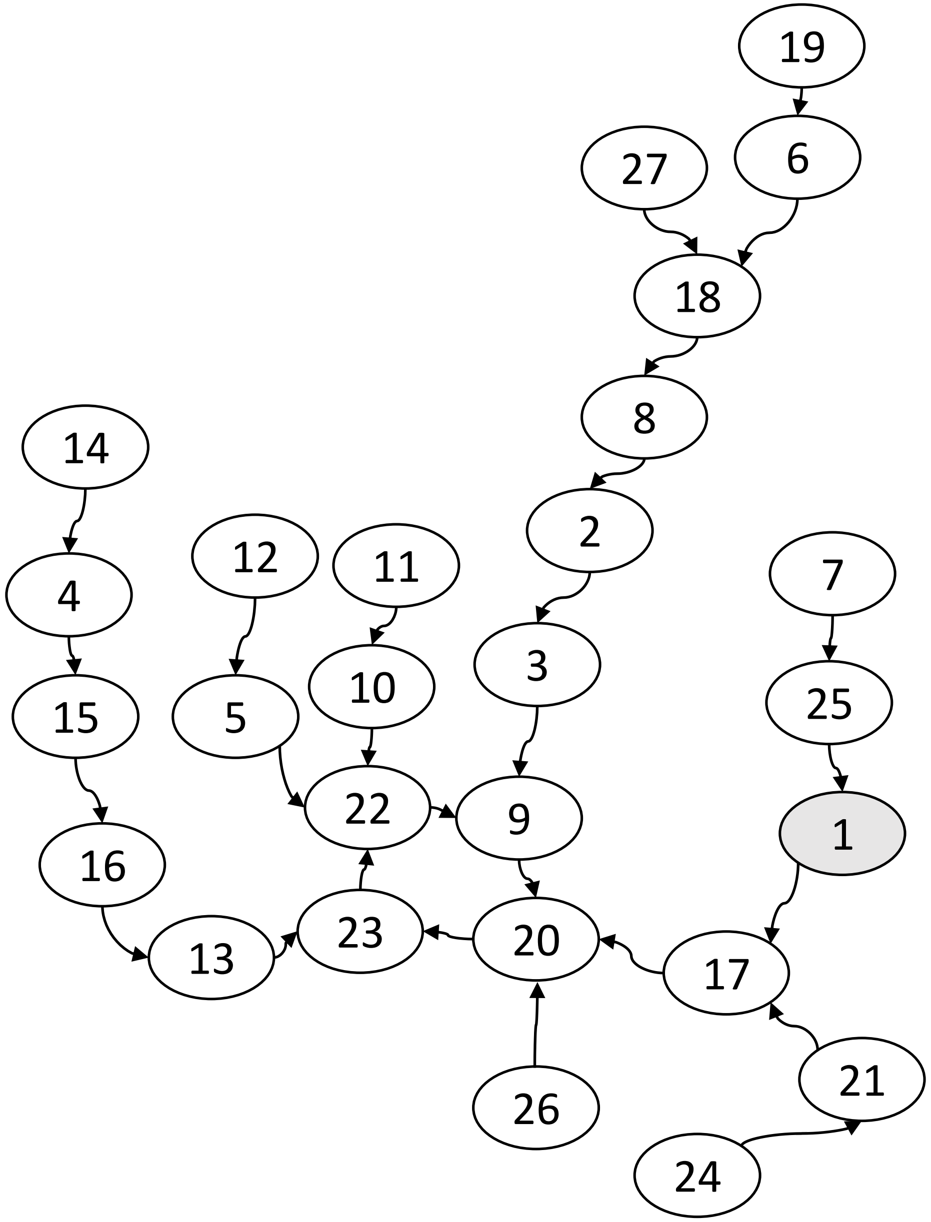


満点解法

- 以下、Bが頂点1と同じ連結成分に属する場合を考える

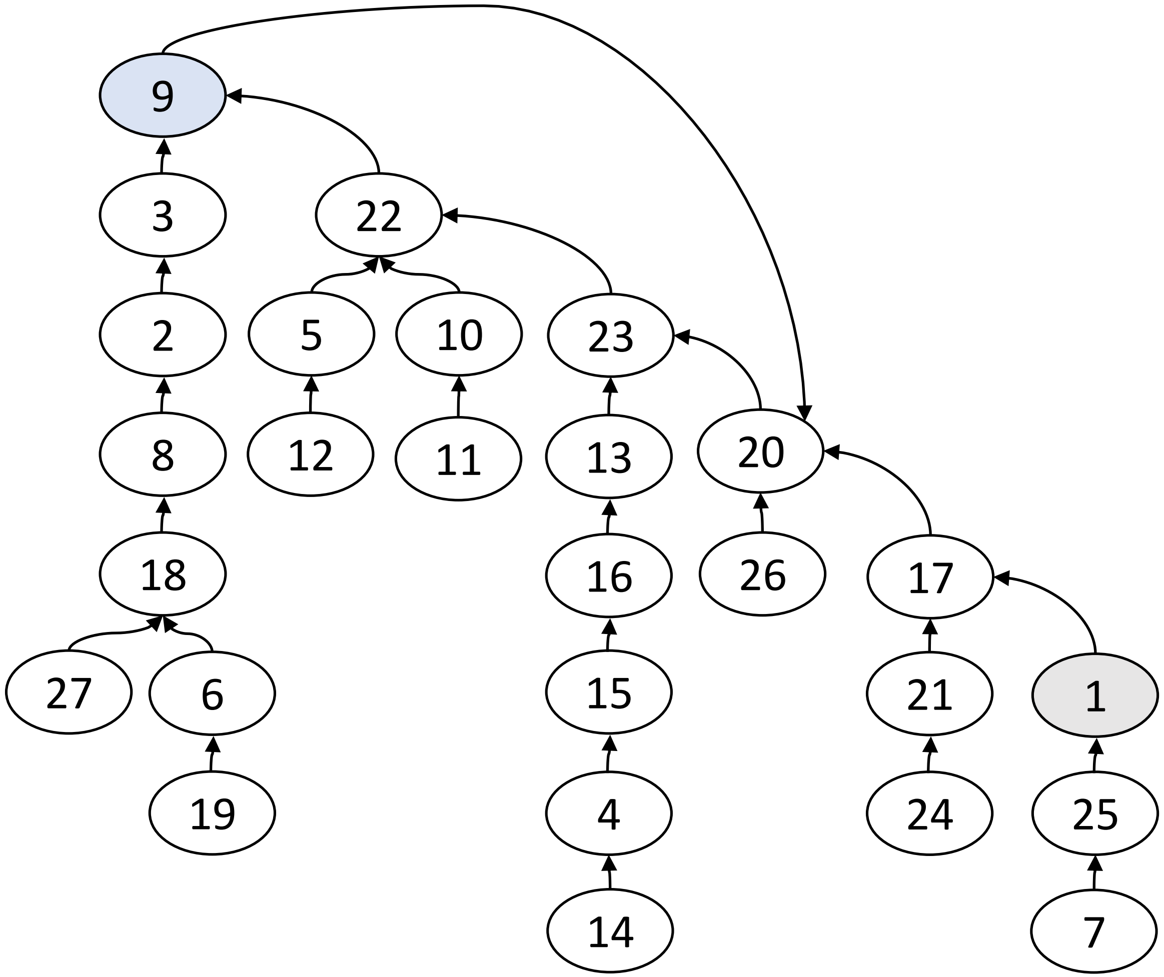
満点解法 – 場合分け(3)

- ここで使うグラフ(N=27, K=35)



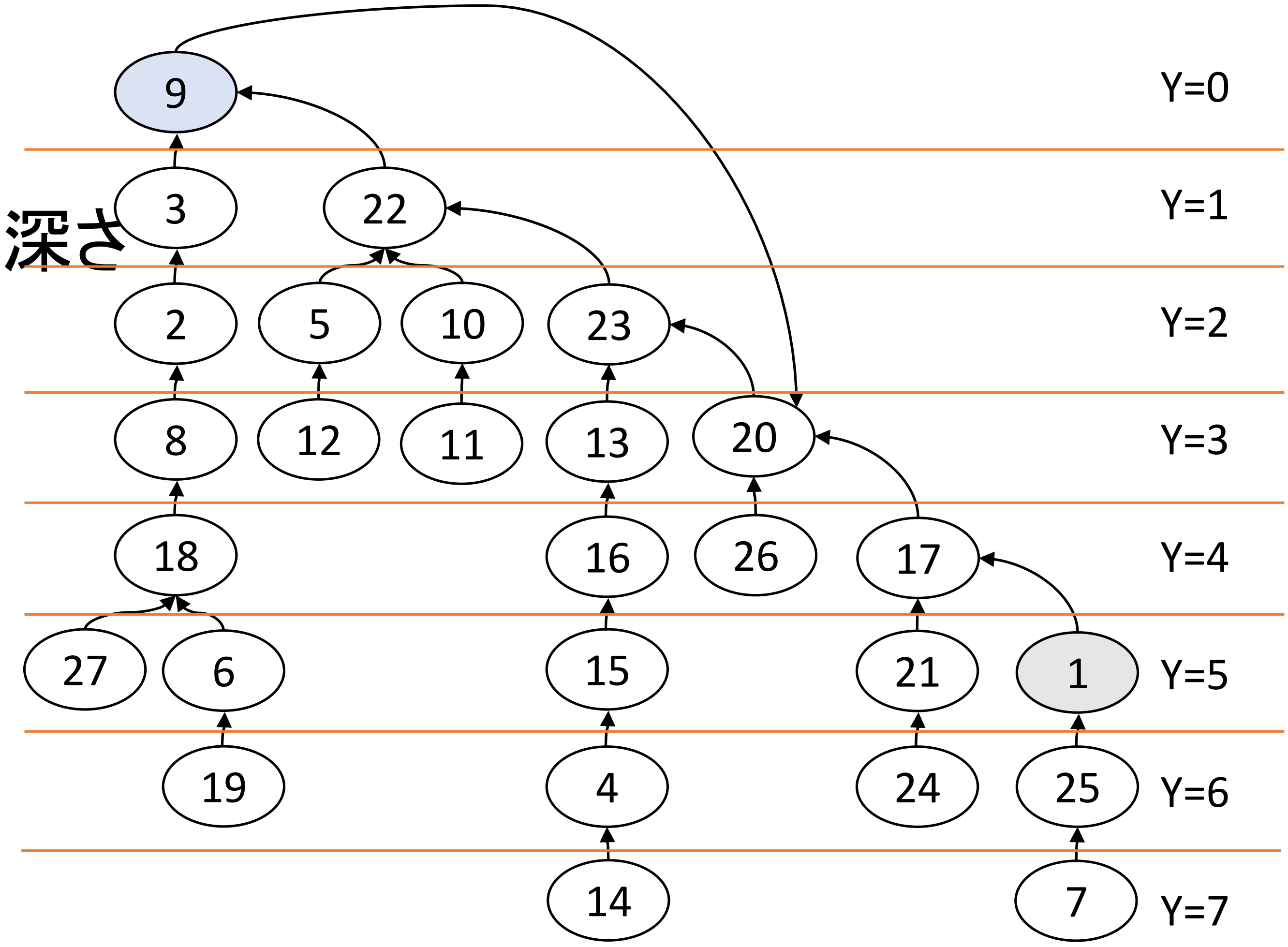
満点解法 – 場合分け(3)

- 根を決める
- 根 = 頂点1から辿ってはじめてループに入る直前の頂点



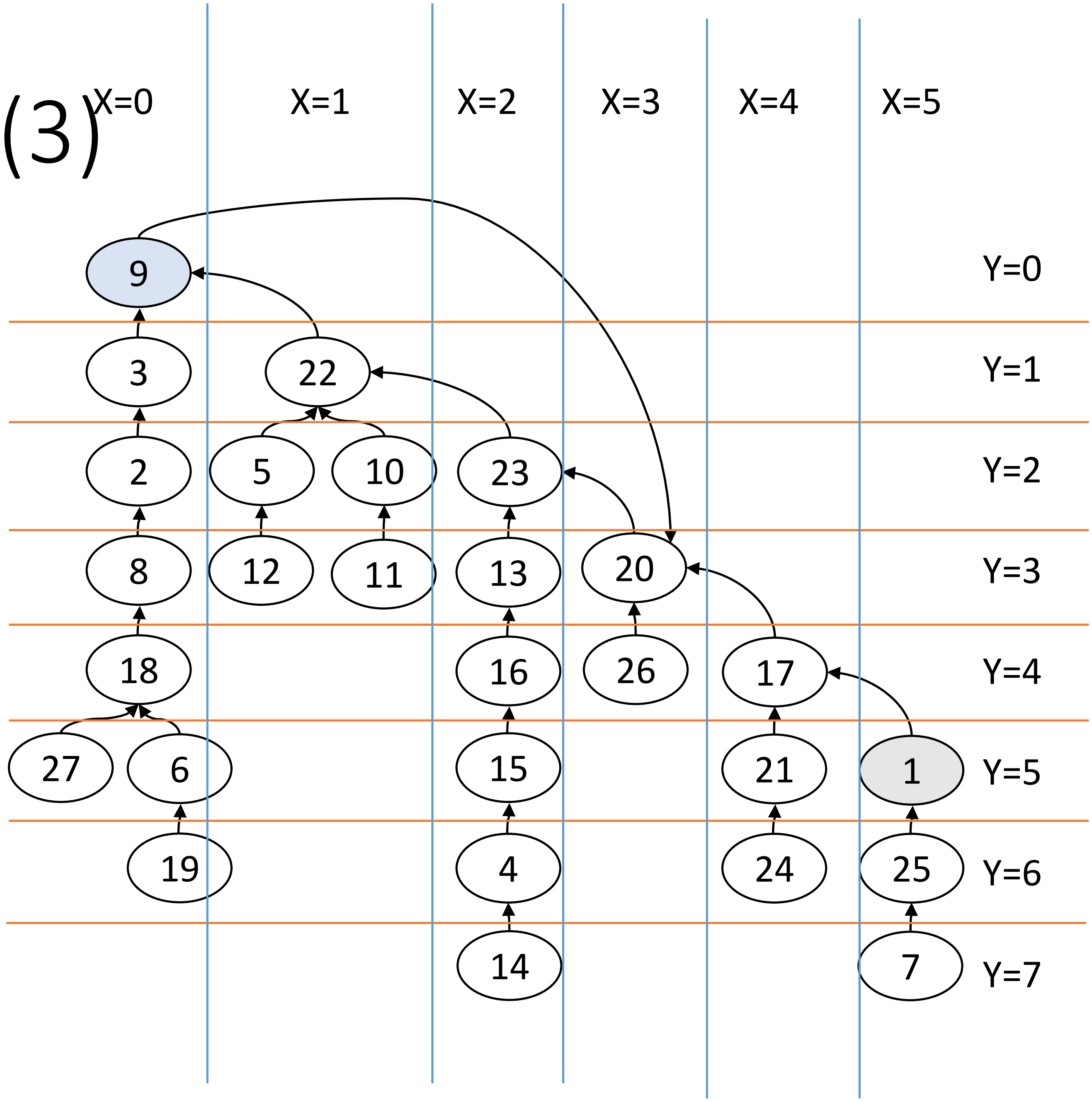
満点解法 – 場合分け(3)

- Y座標を定める
- Y座標 : これを木として見た時の深さ



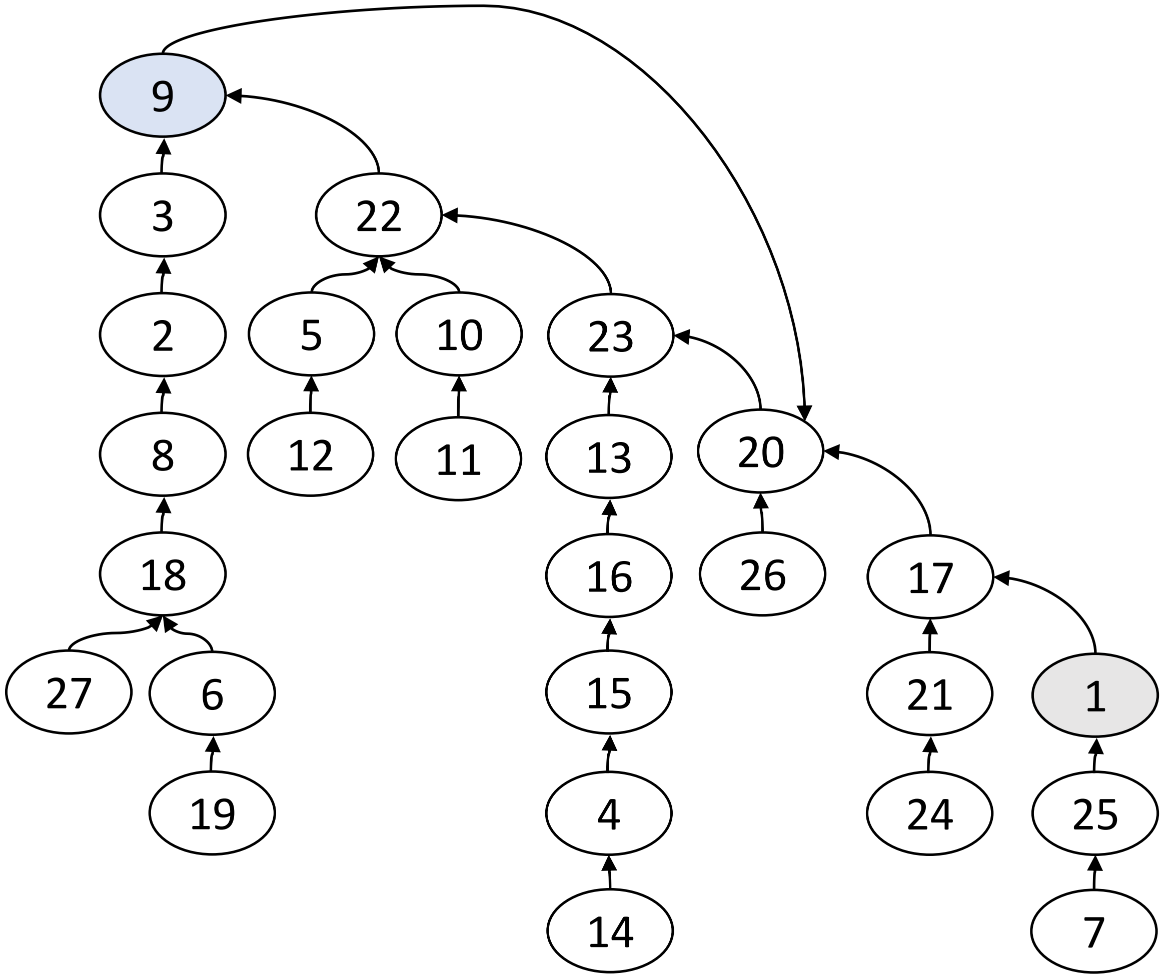
満点解法 – 場合分け(3)

- X座標を定める
- X座標：
  頂点1とのLCAのY座標



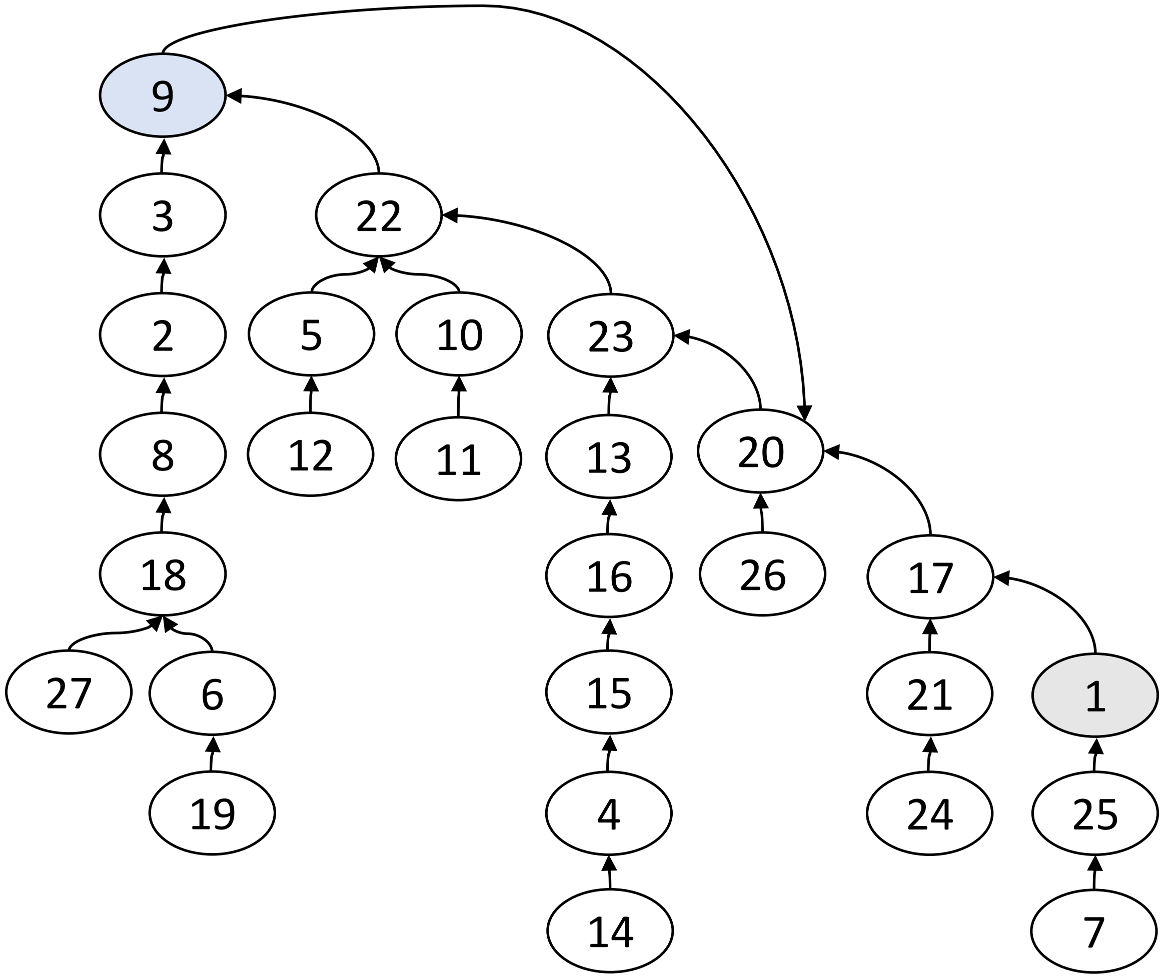
満点解法 – 場合分け(3)

- Dを次の値とする:
  (BのY座標) – (AのY座標)
- Lを次の値とする:
  (初期状態のループ長)



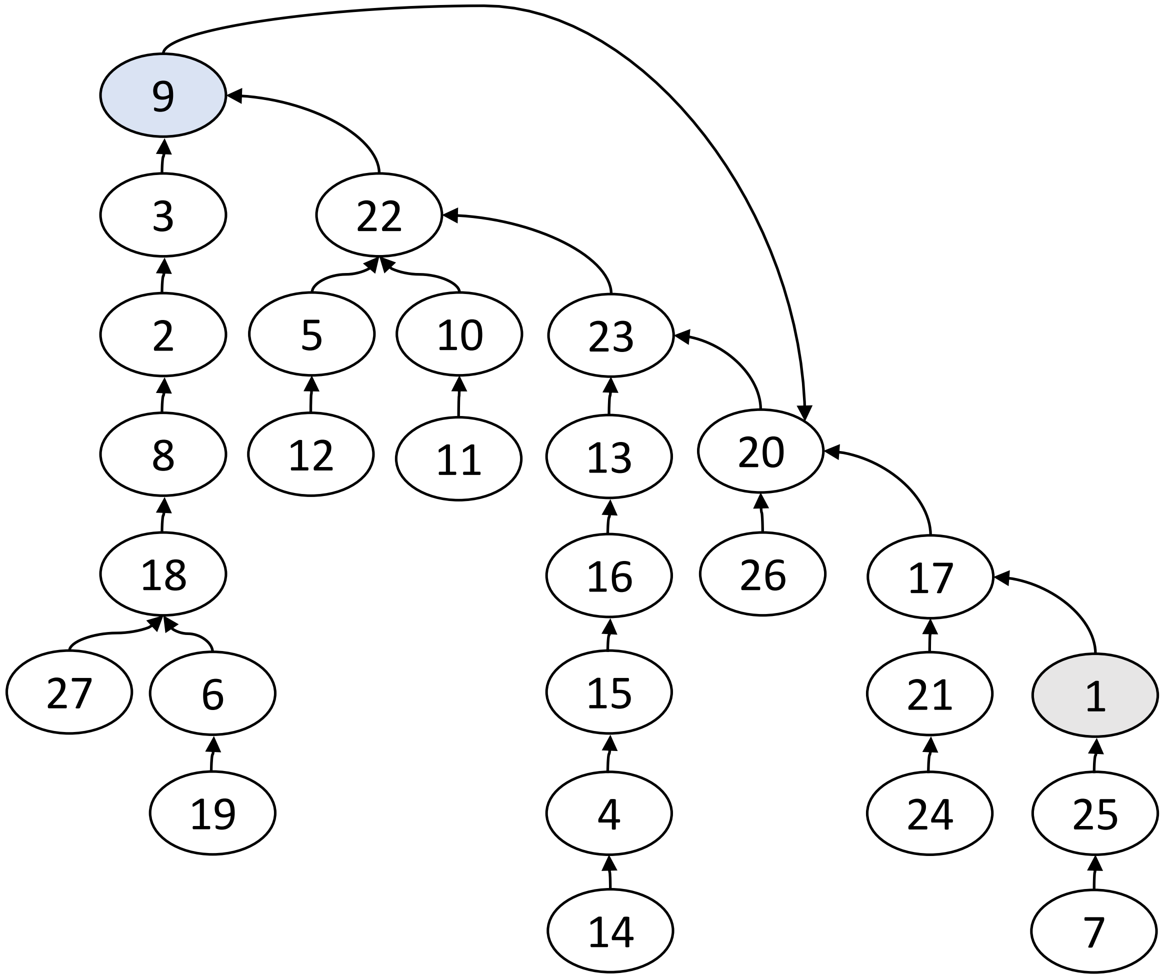
満点解法 – 場合分け(3)

- Dを次の値とする:
  (BのY座標) – (AのY座標)
- Lを次の値とする:
  (初期状態のループ長)
- この例ではL=4
  (DはA,Bの選び方に依存する値)



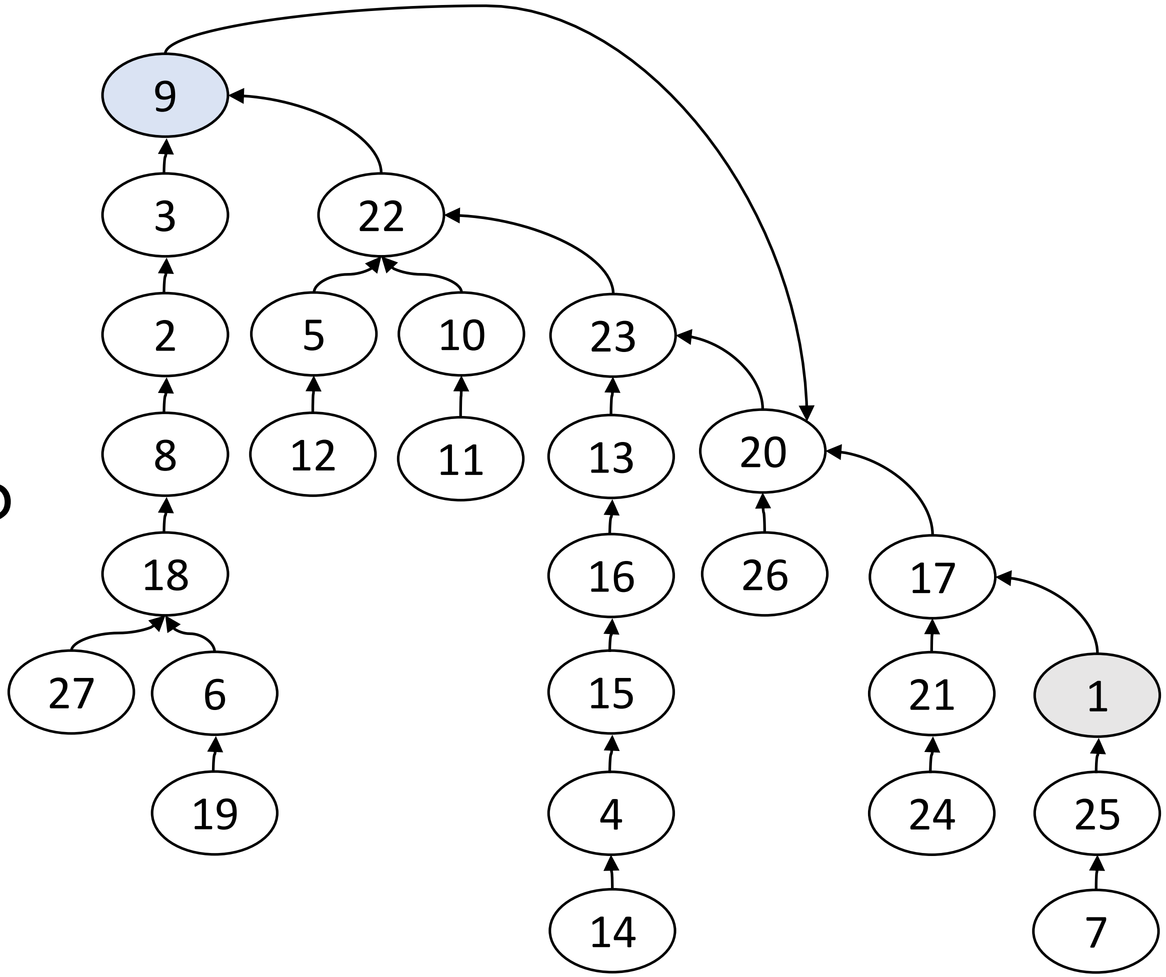
満点解法 – 場合分け(3)

- AがBの祖先でない場合をまず考える



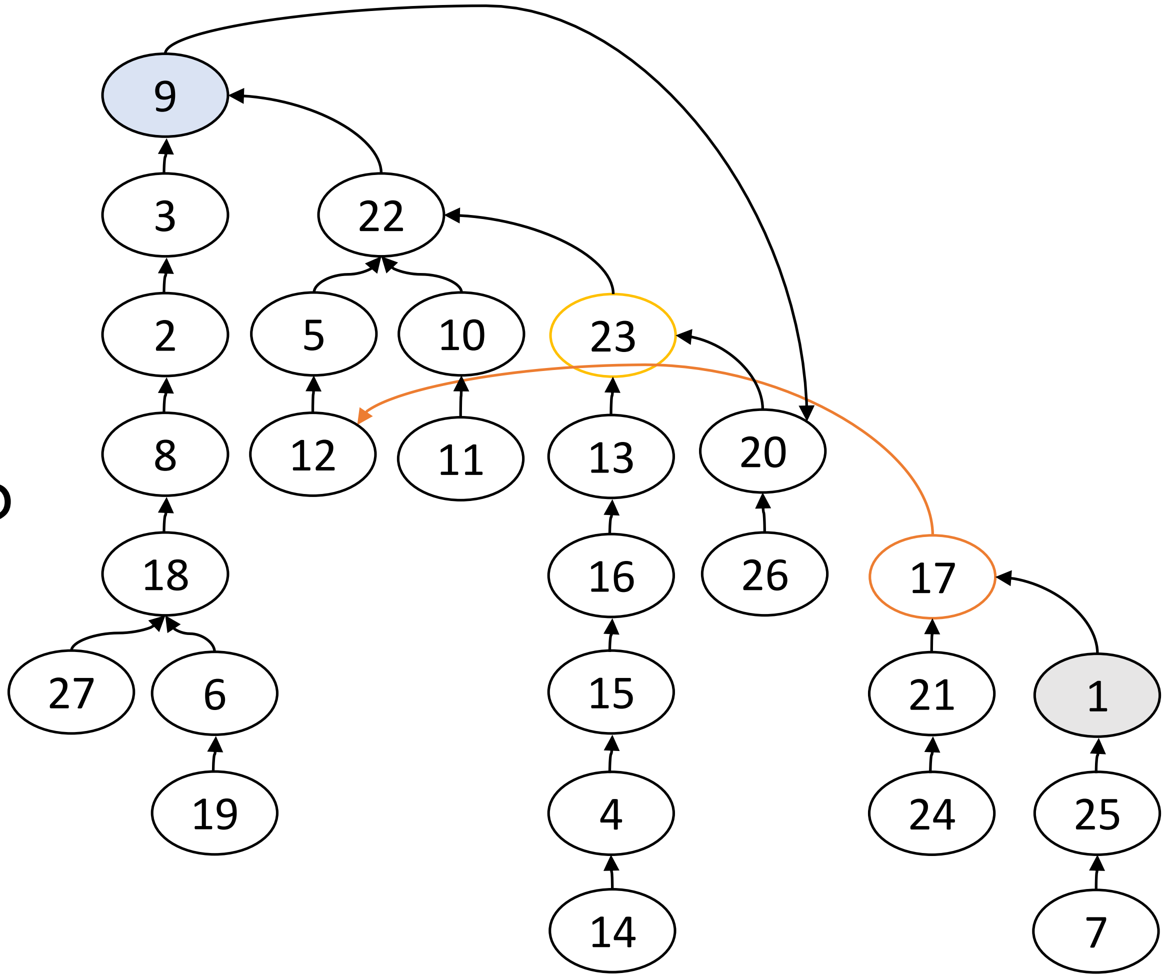
満点解法 – 場合分け(3)

- AがBの祖先でない場合をまず考える
- 特に、Aがサイクルに含まれない場合(この例ではA=1,17)をまず考える



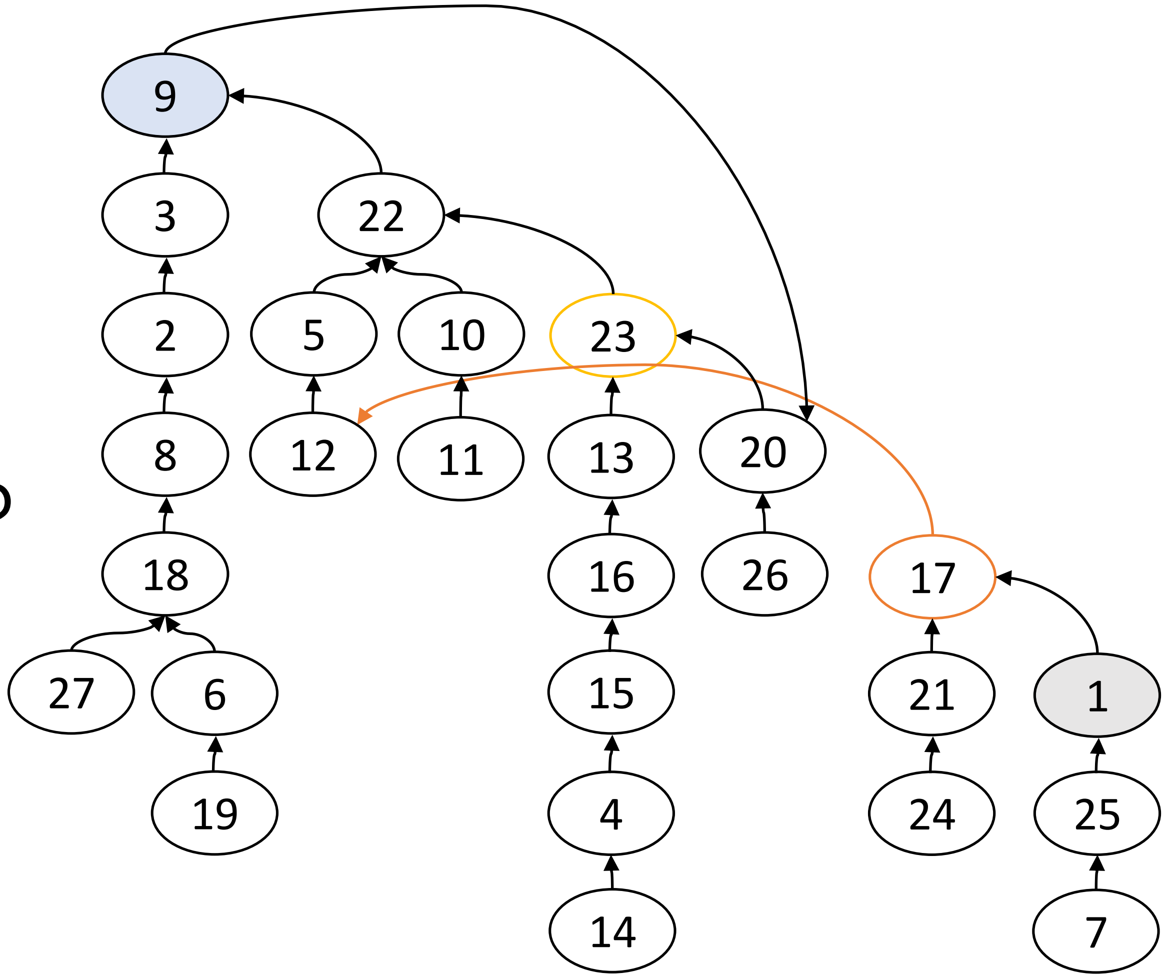
満点解法 – 場合分け(3A)

- AがBの祖先でない場合を
まず考える
- 特に、Aがサイクルに
含まれない場合
(この例ではA=1,17)をまず考える



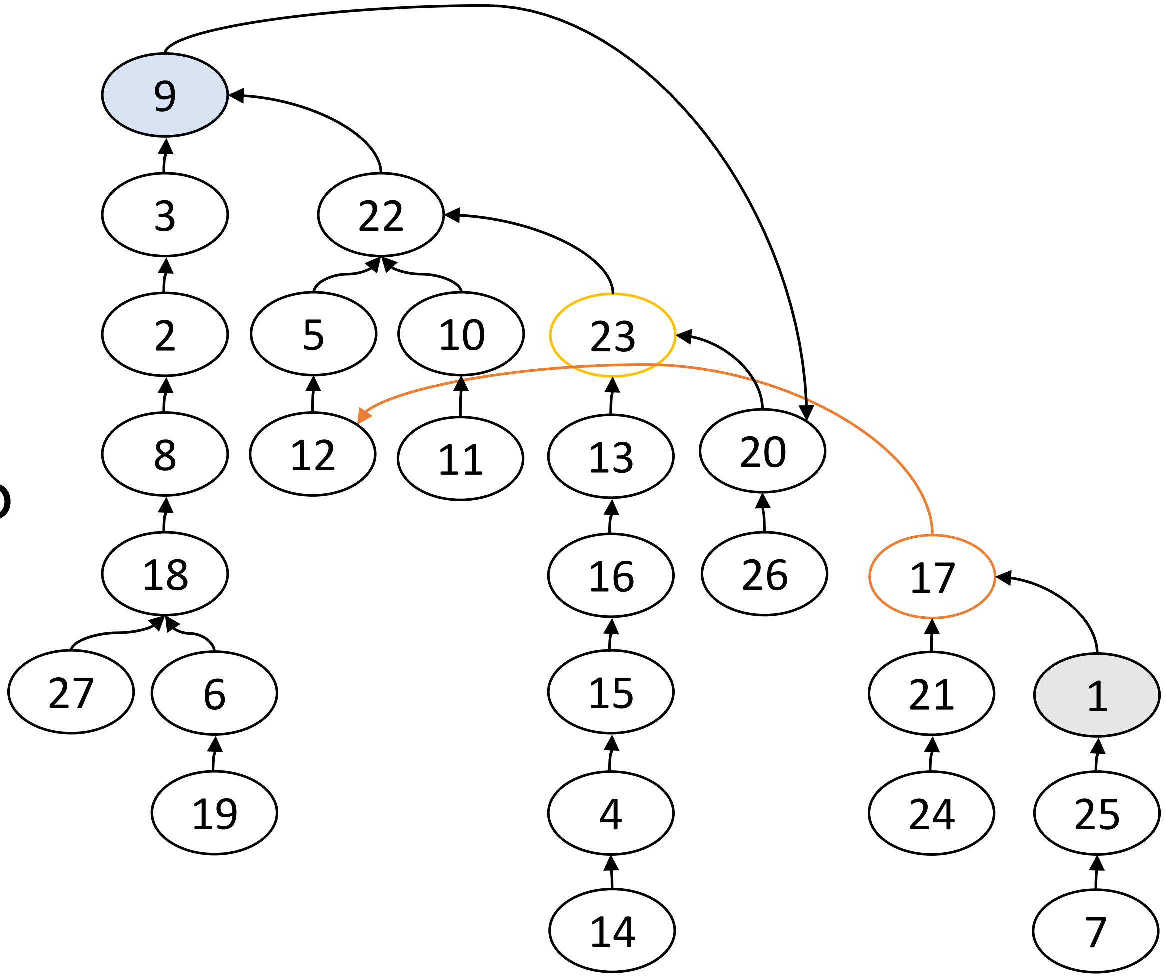
満点解法 – 場合分け(3A)

- AがBの祖先でない場合をまず考える
- 特に、Aがサイクルに含まれない場合(この例ではA=1,17)をまず考える
- このときはサイクル長は一定



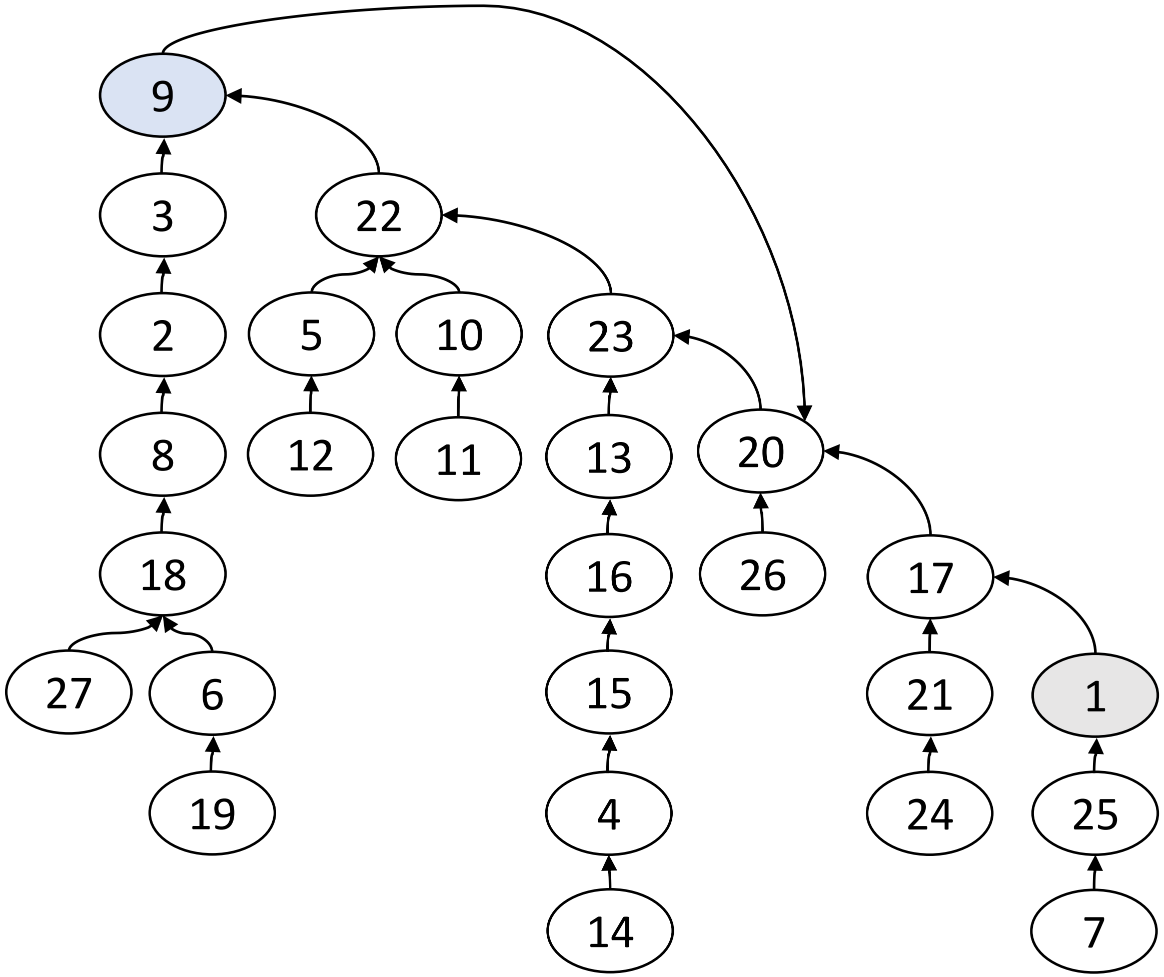
満点解法 – 場合分け(3A)

- AがBの祖先でない場合をまず考える
- 特に、Aがサイクルに含まれない場合(この例ではA=1,17)をまず考える
- このときはサイクル長は一定
- 場合分け(2)と同様に処理可能



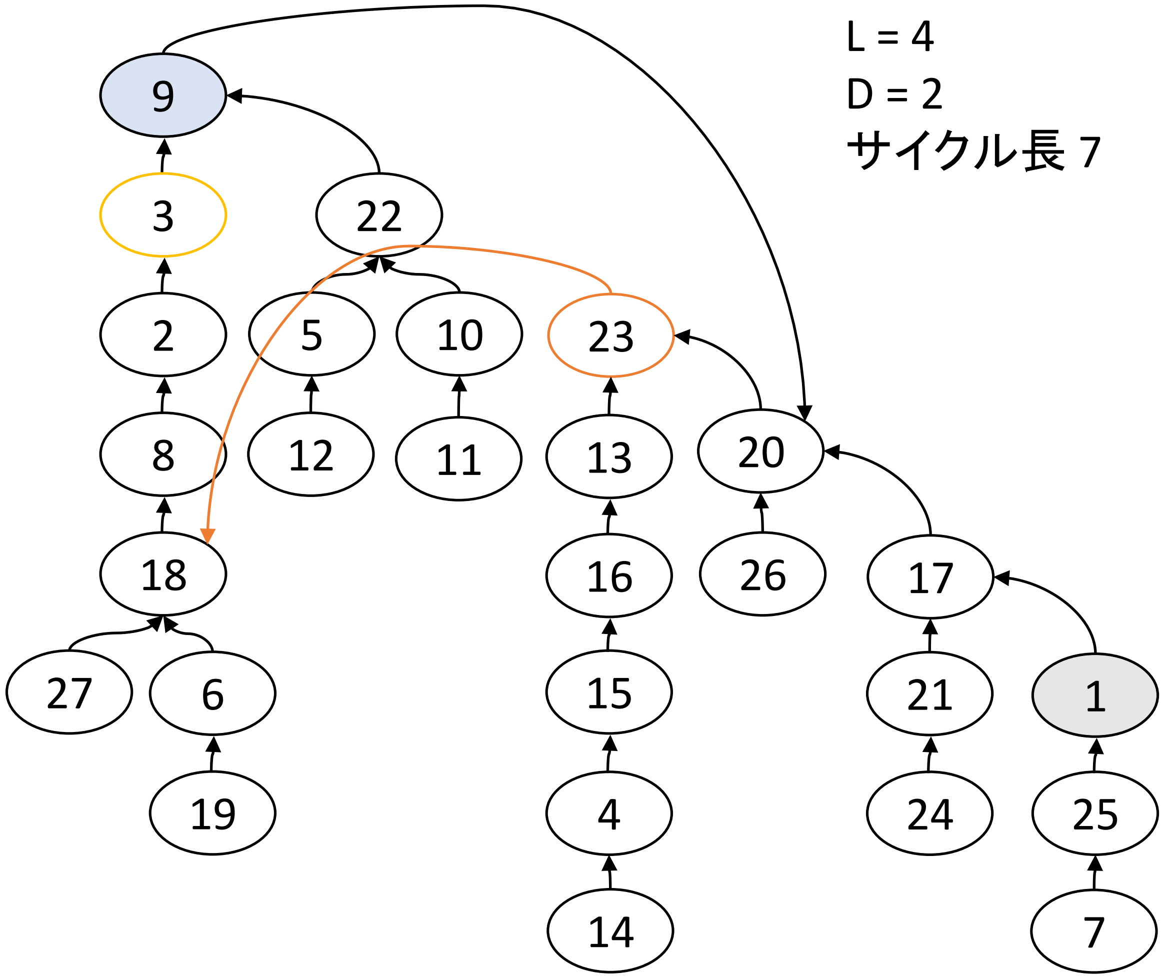
満点解法 – 場合分け(3)

- AがBの祖先でない場合をまず考える
- 次:Aがサイクルに含まれる場合(この例ではA=20,23,22,9)



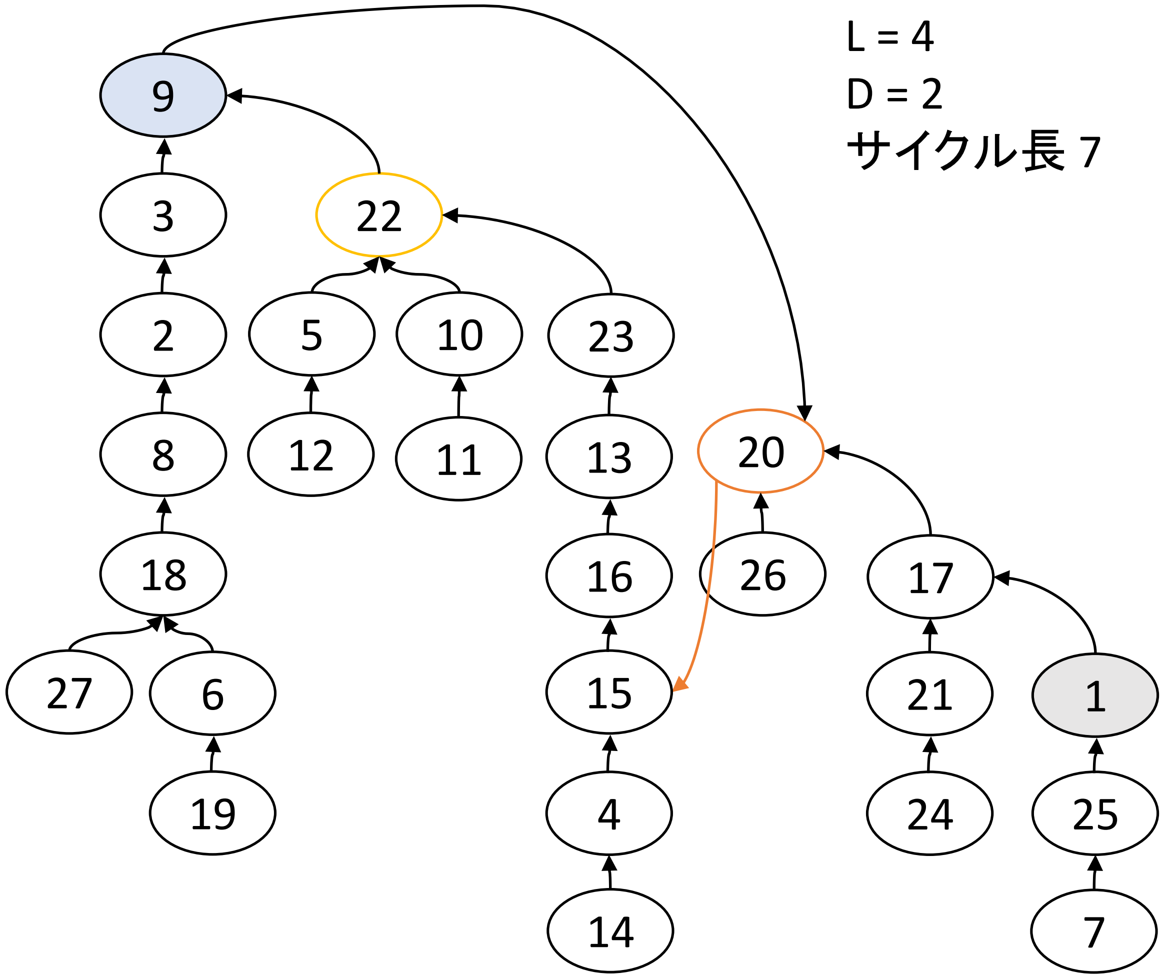
満点解法 – 場合分け(3B)

- AがBの祖先でない場合をまず考える
- 次:Aがサイクルに含まれる場合(この例ではA=20,23,22,9)
- このとき、できるサイクル長はL+D+1である。



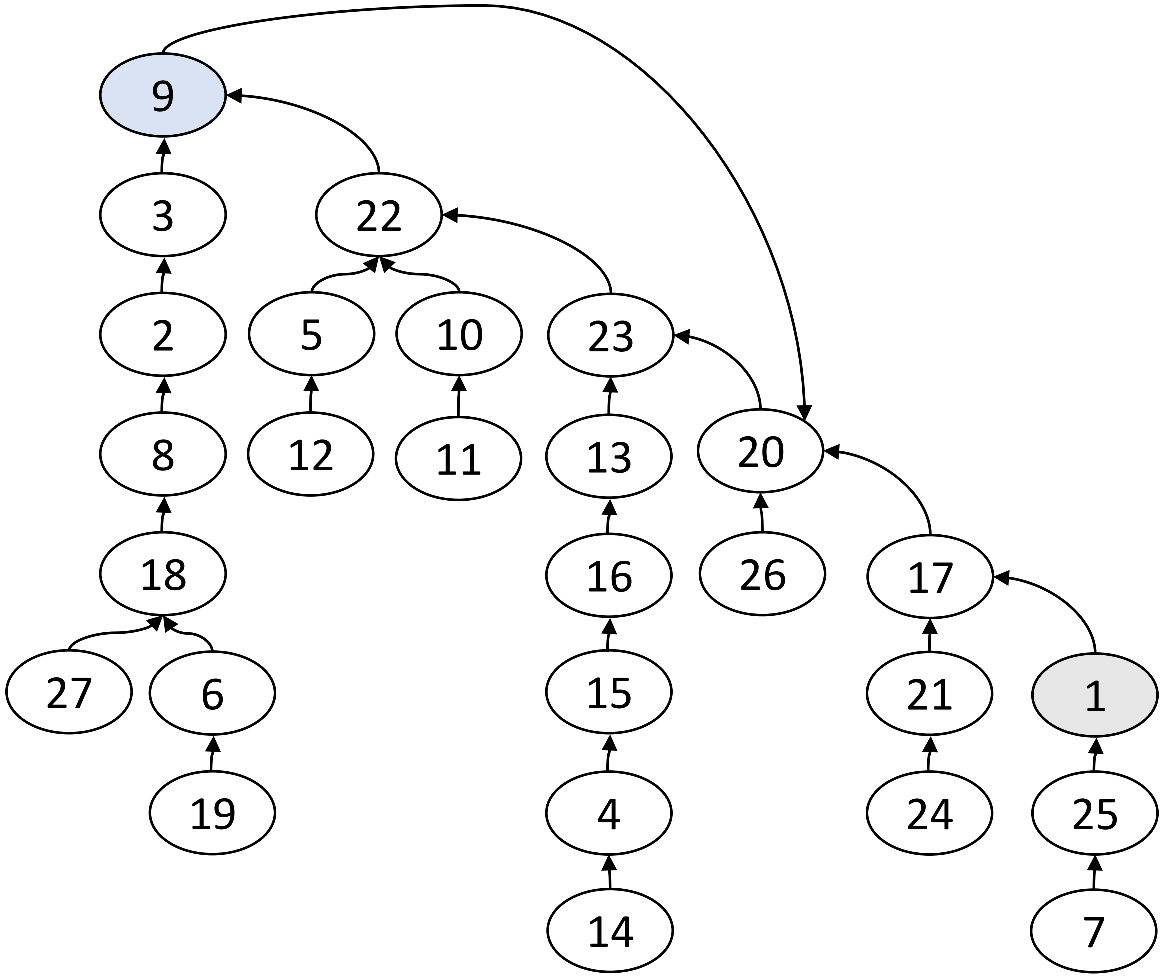
満点解法 – 場合分け(3B)

- AがBの祖先でない場合をまず考える
- 次:Aがサイクルに含まれる場合(この例ではA=20,23,22,9)
- このとき、できるサイクル長はL+D+1である。
- D一定のとき、到達先のY座標は一定…？



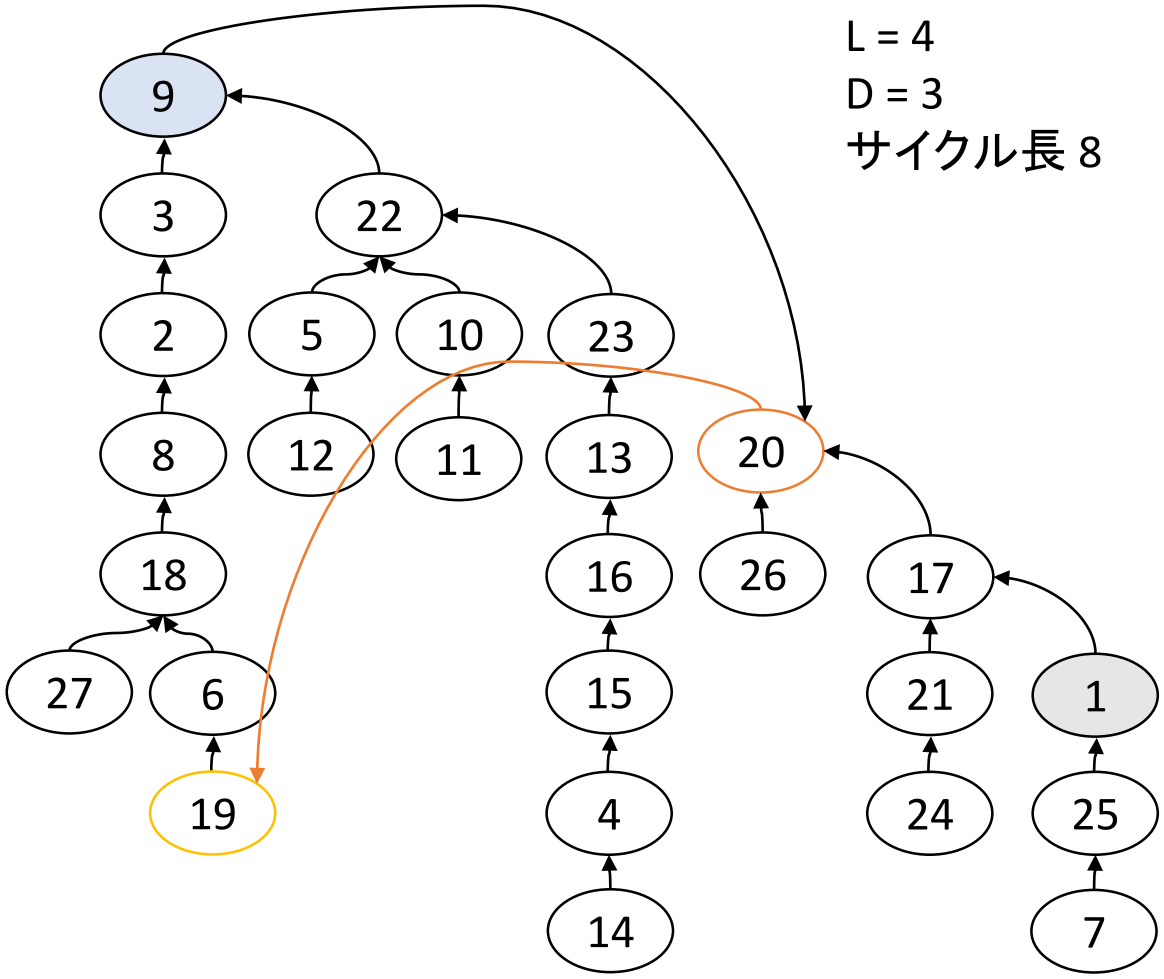
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない



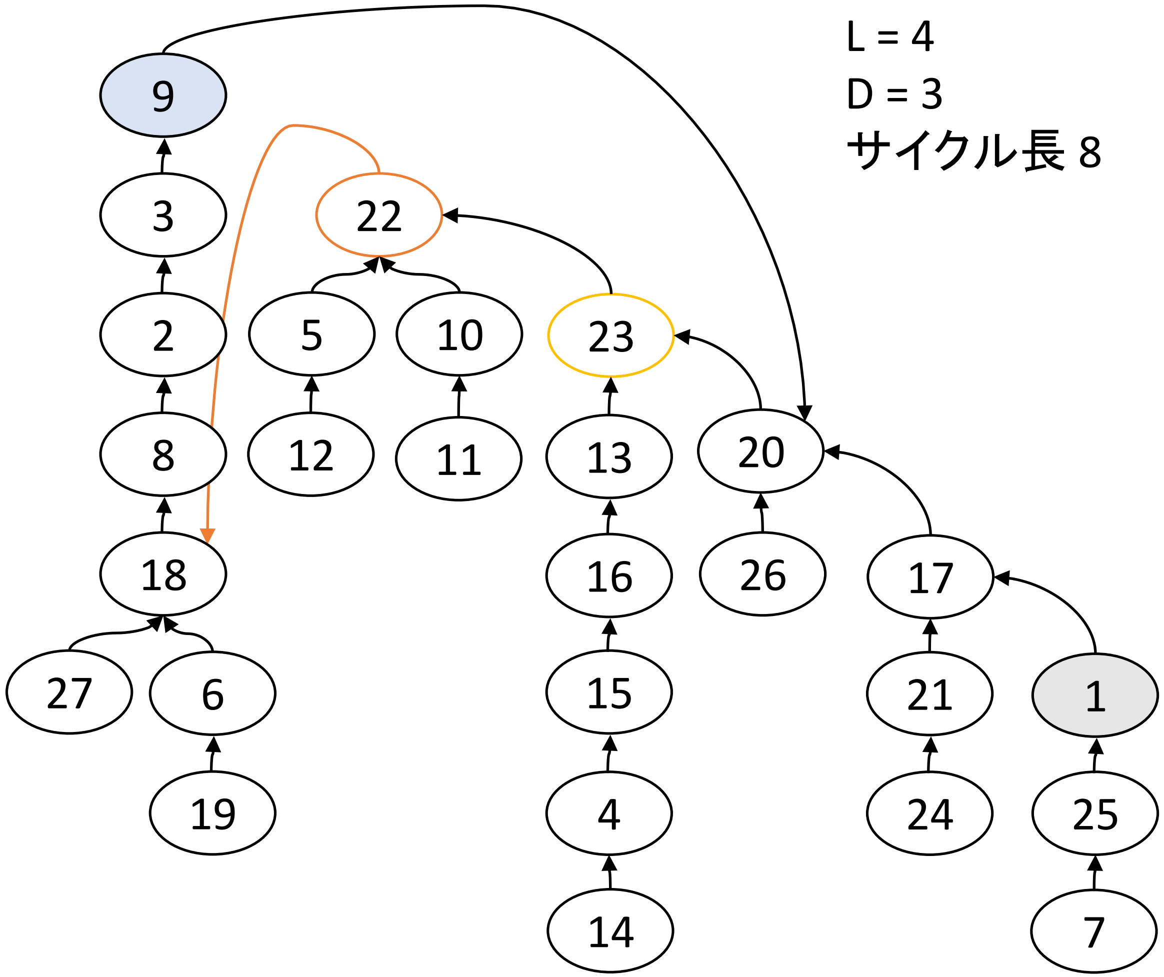
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない
- このときはY=6



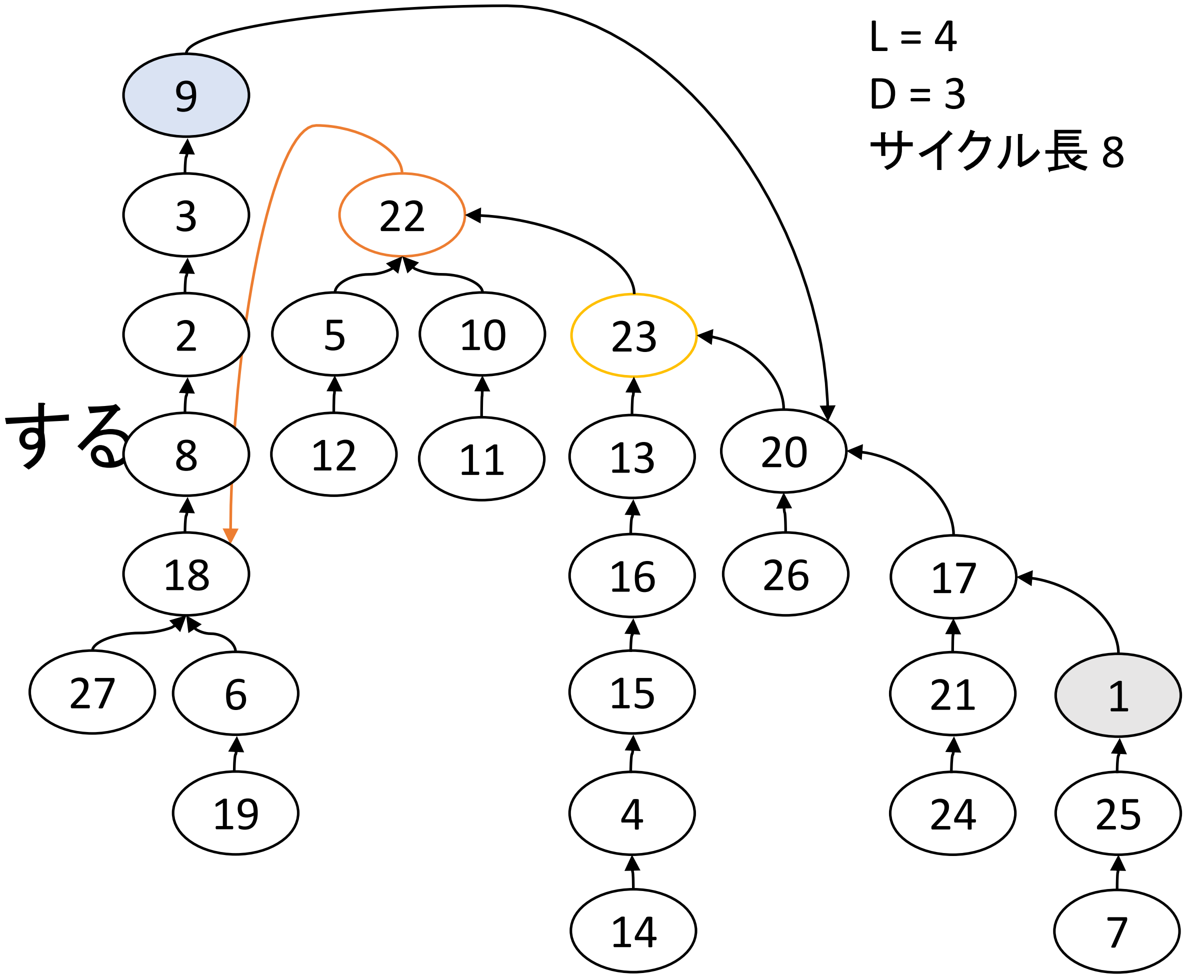
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない
- このときはY=2となる



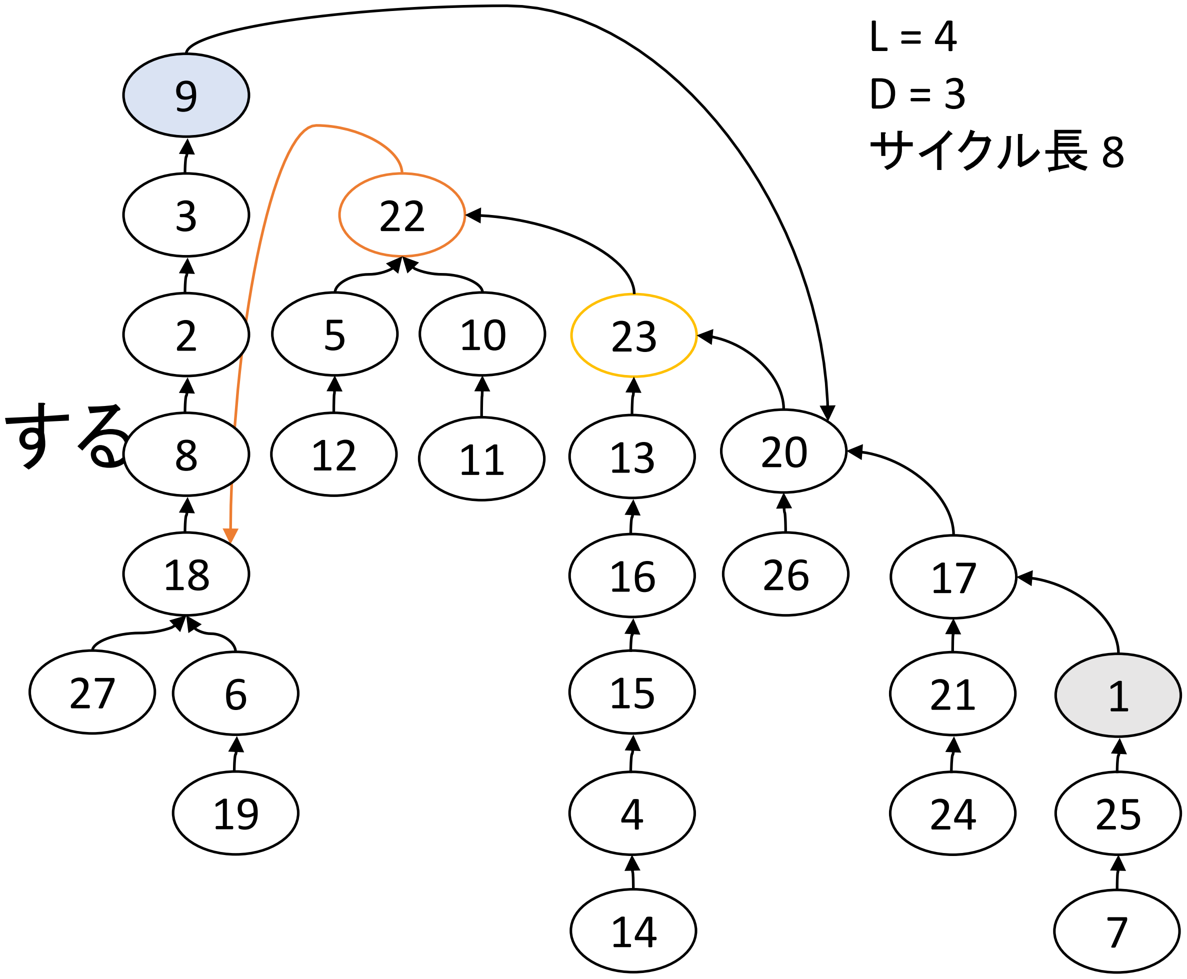
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない
- このときはY=2となる
- このように一点だけ例外が存在する



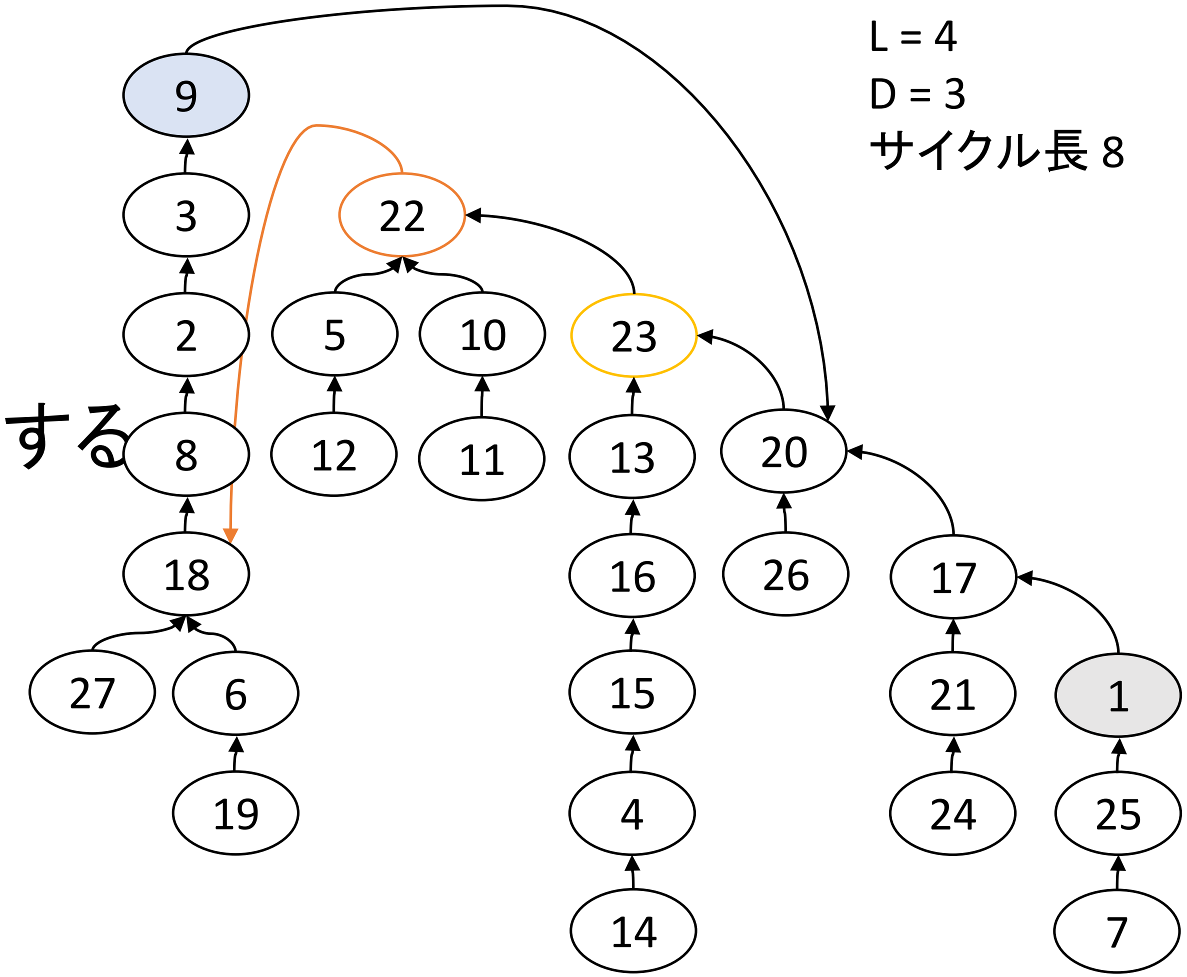
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない
- このときはY=2となる
- このように一点だけ例外が存在する
- この例外はループ上に存在



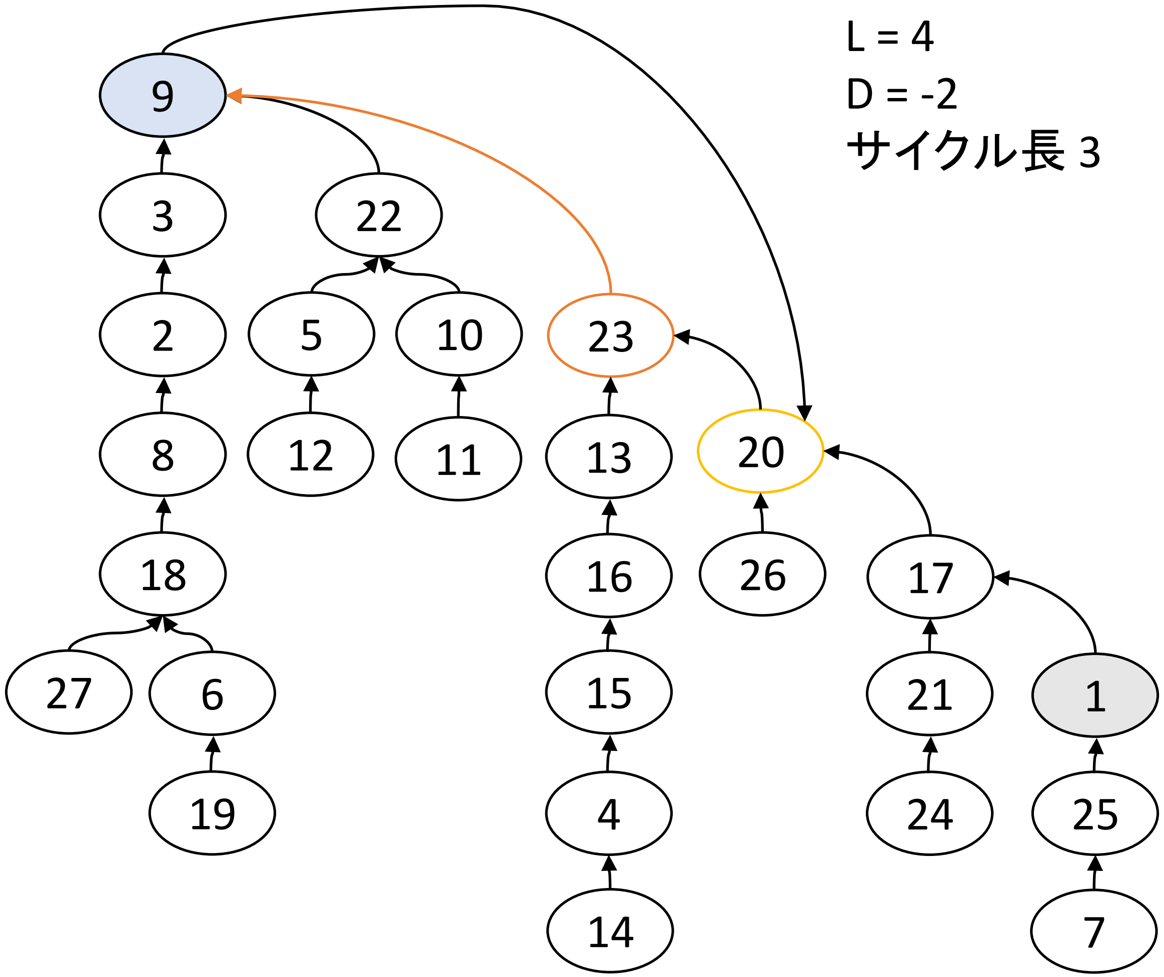
満点解法 – 場合分け(3B)

- D一定のとき、到達先のY座標は一定ではない
- このときはY=2となる
- このように一点だけ例外が存在する
- この例外はループ上に存在
- この座標は通常のもののY座標からD+1を引いたものとなる



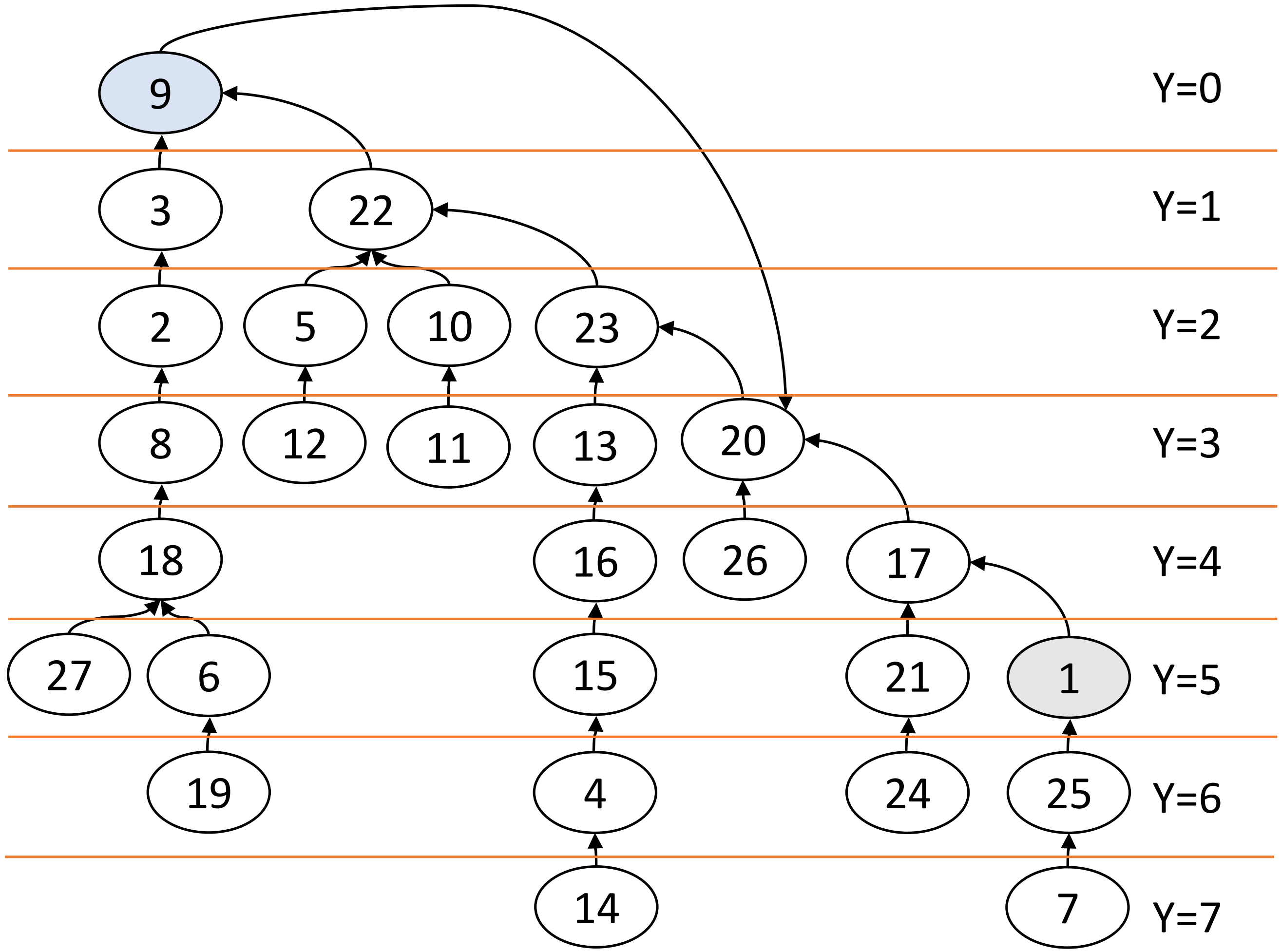
満点解法 – 場合分け(3B)

- Dはマイナスにもなるので注意



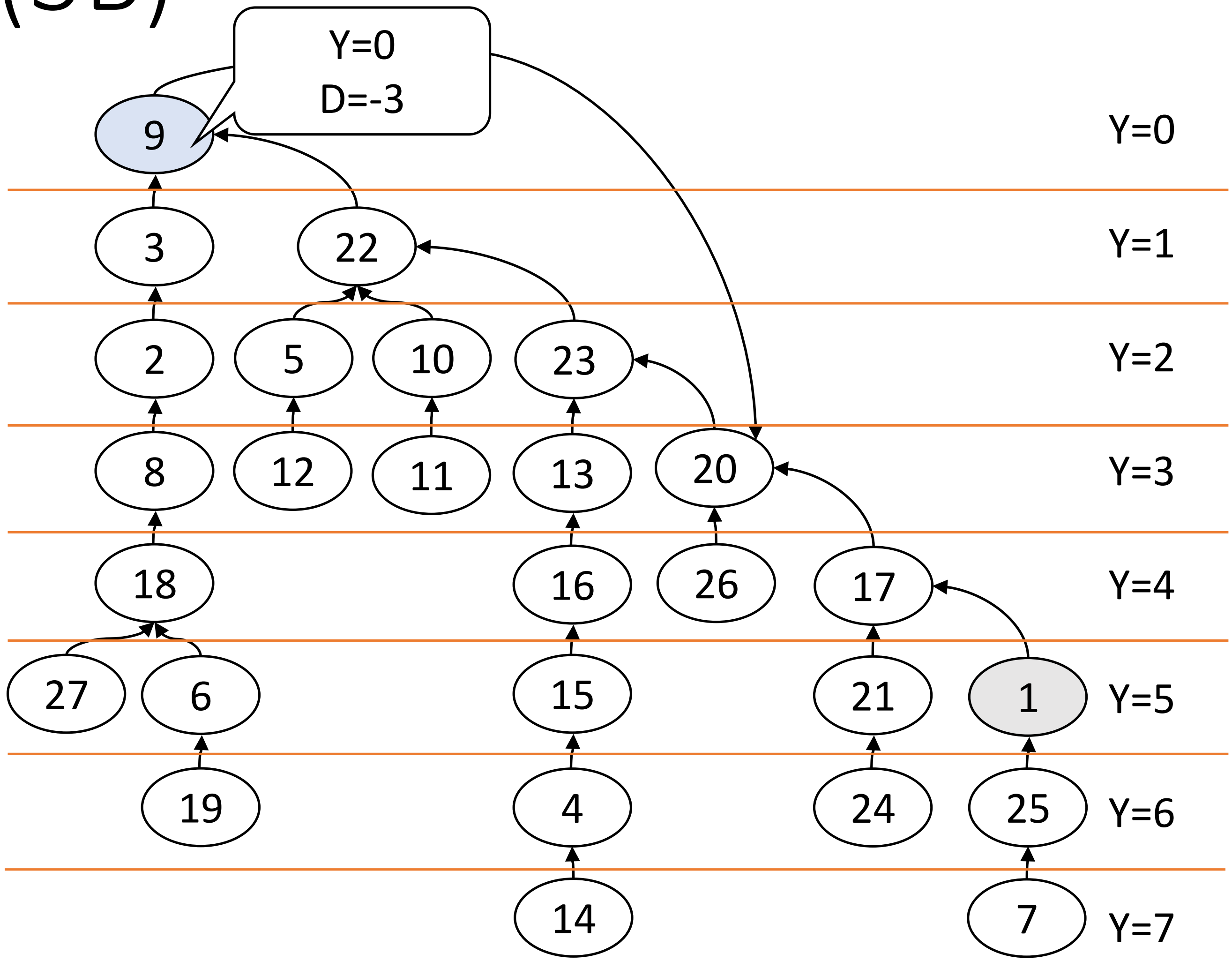
満点解法 – 場合分け(3B)

- 以上をイベントとして管理



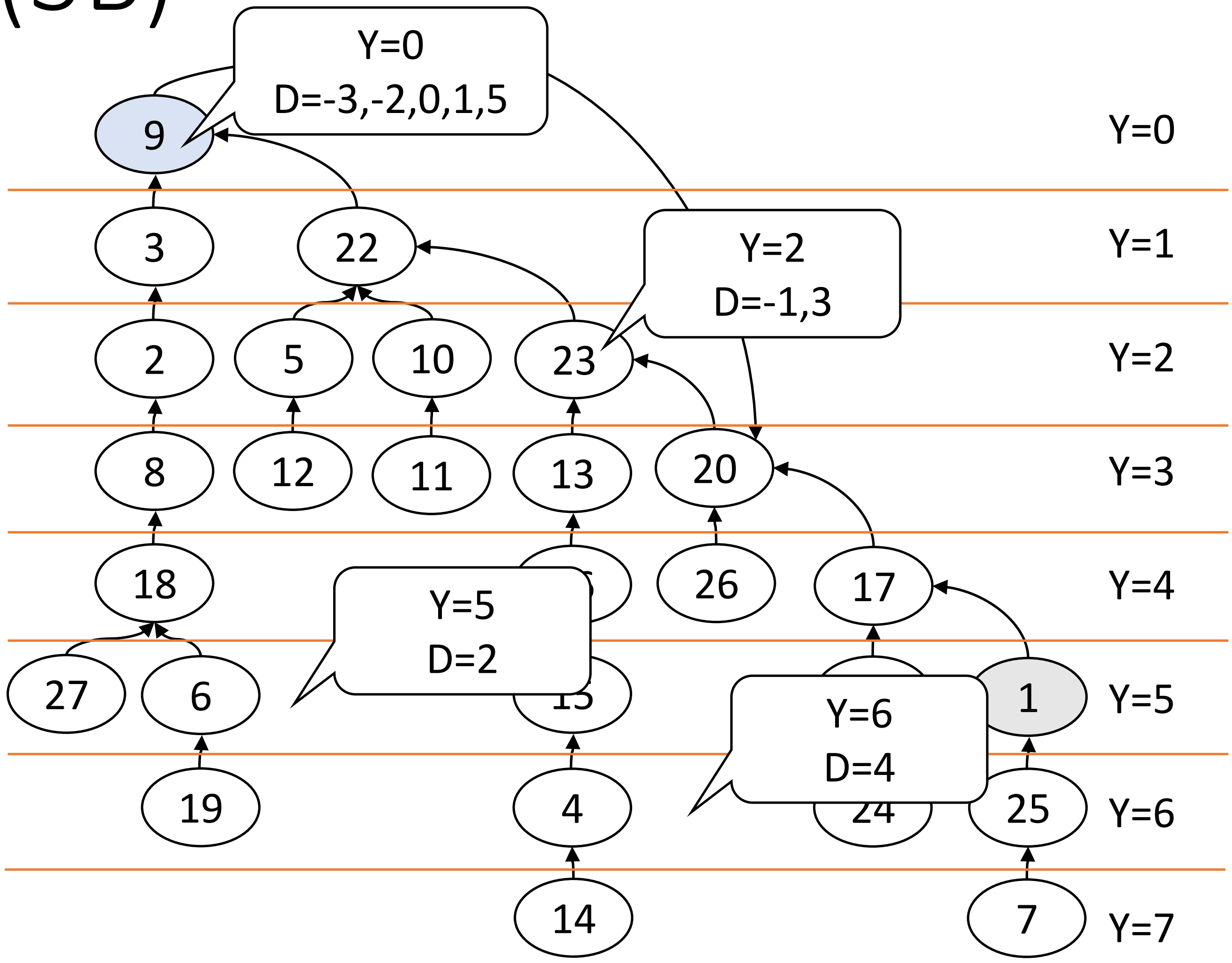
満点解法 – 場合分け(3B)

- 以上をイベントとして管理



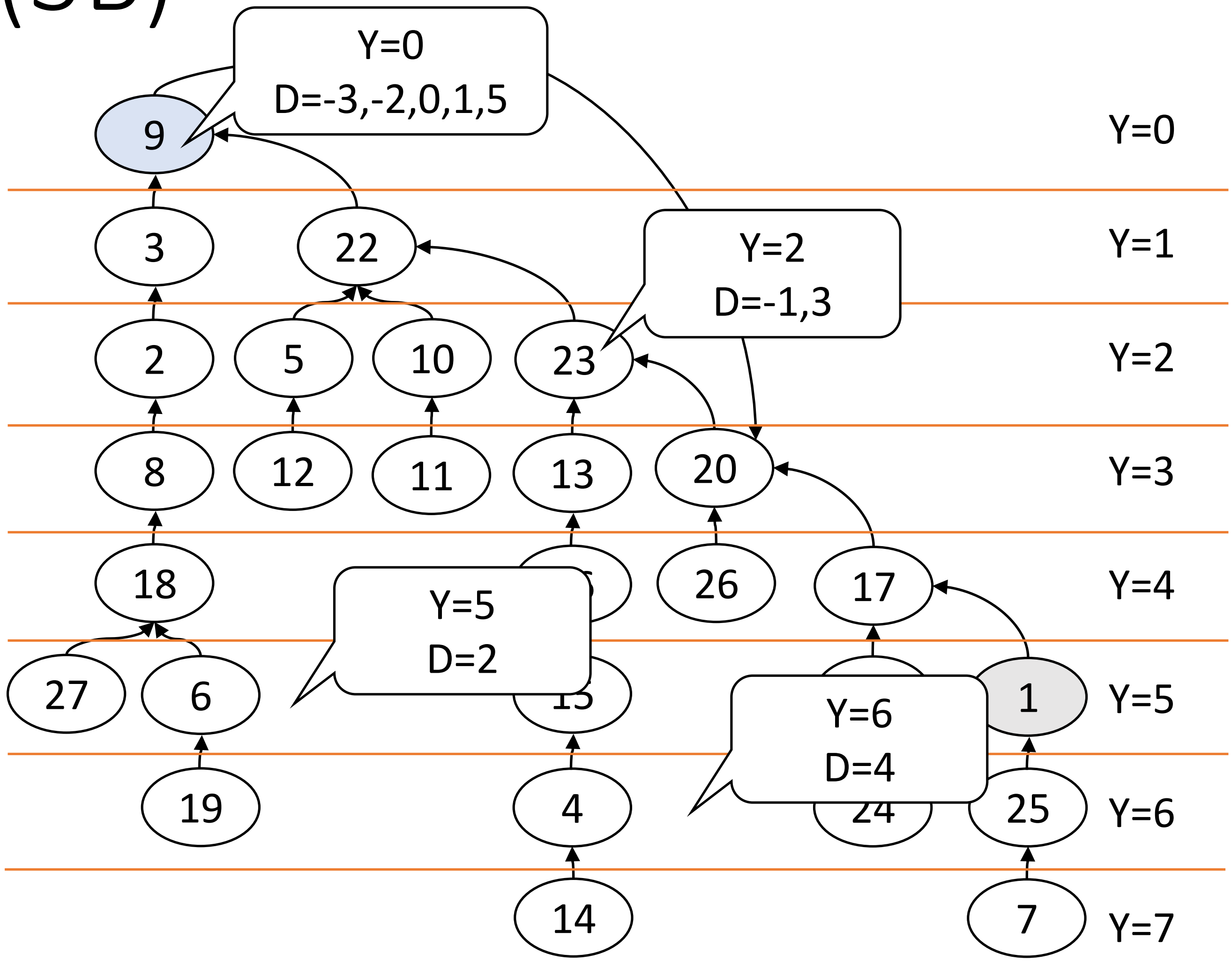
満点解法 – 場合分け(3B)

- 以上をイベントとして管理
- 各Dについて、対応するY座標を計算



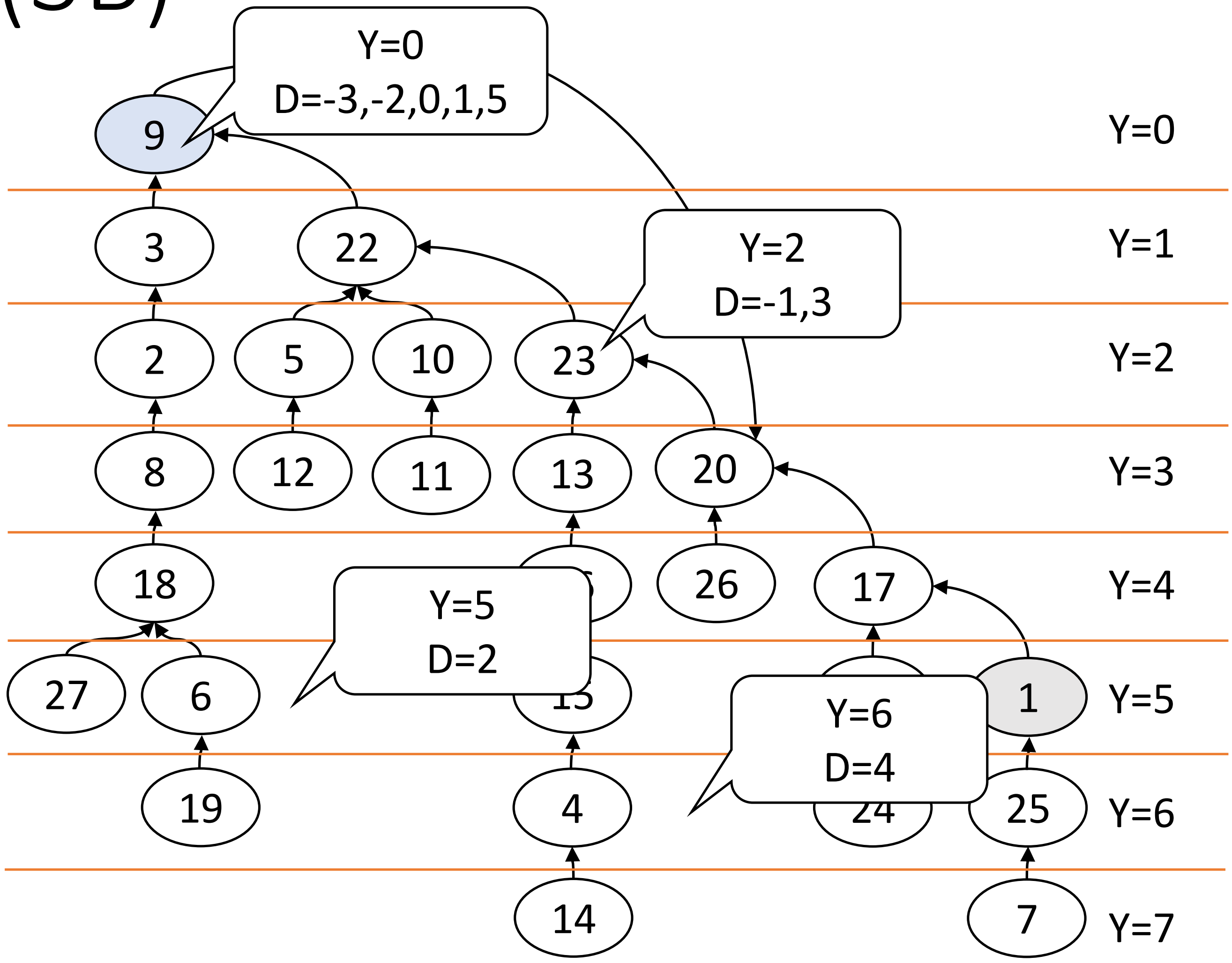
満点解法 – 場合分け(3B)

- 以上をイベントとして管理
- 各Dについて、対応するY座標を計算
- 例外的な到達先のY座標はこの値からD+1を減じたものになる



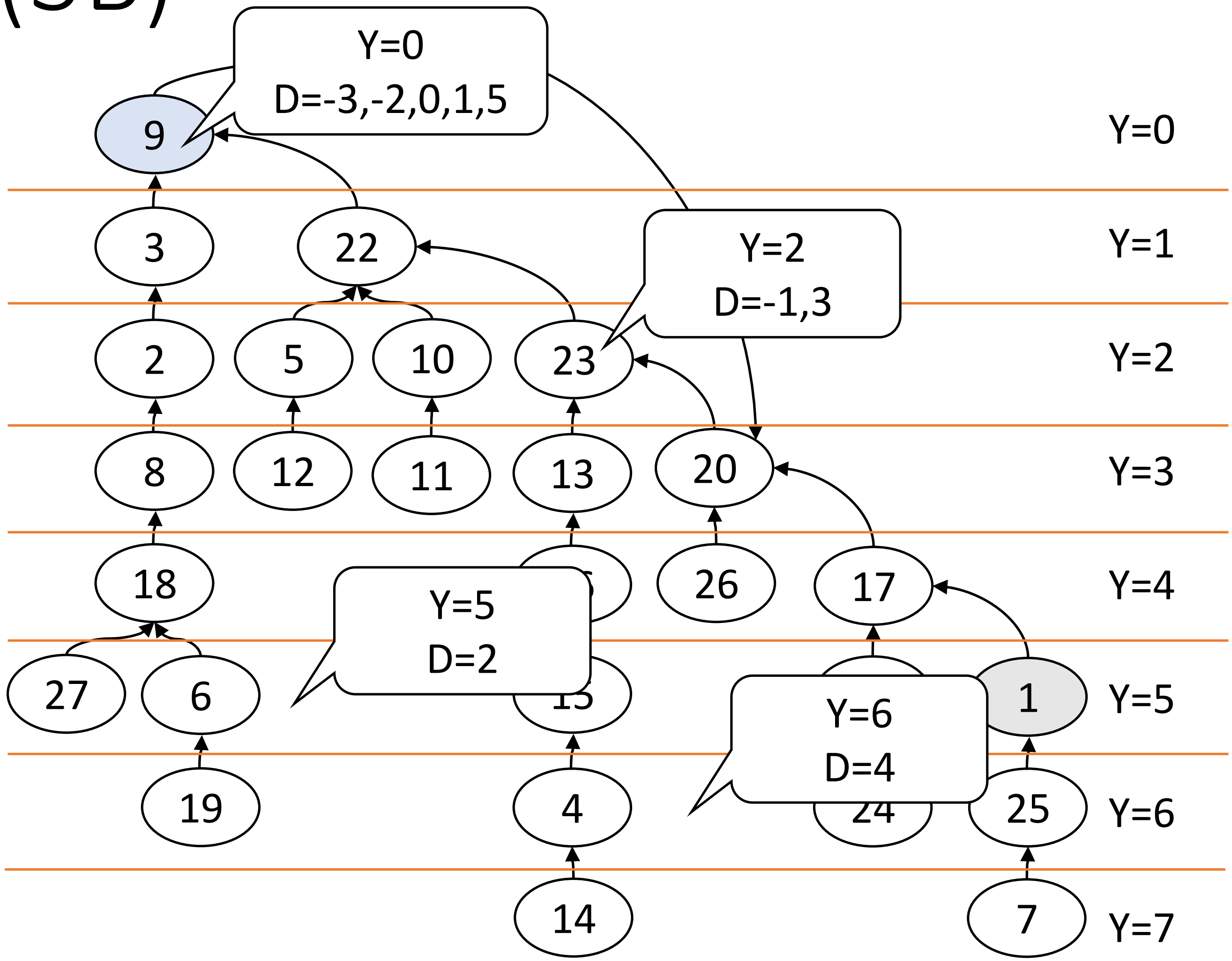
満点解法 – 場合分け(3B)

- 実際の数え上げ:
  下から走査する



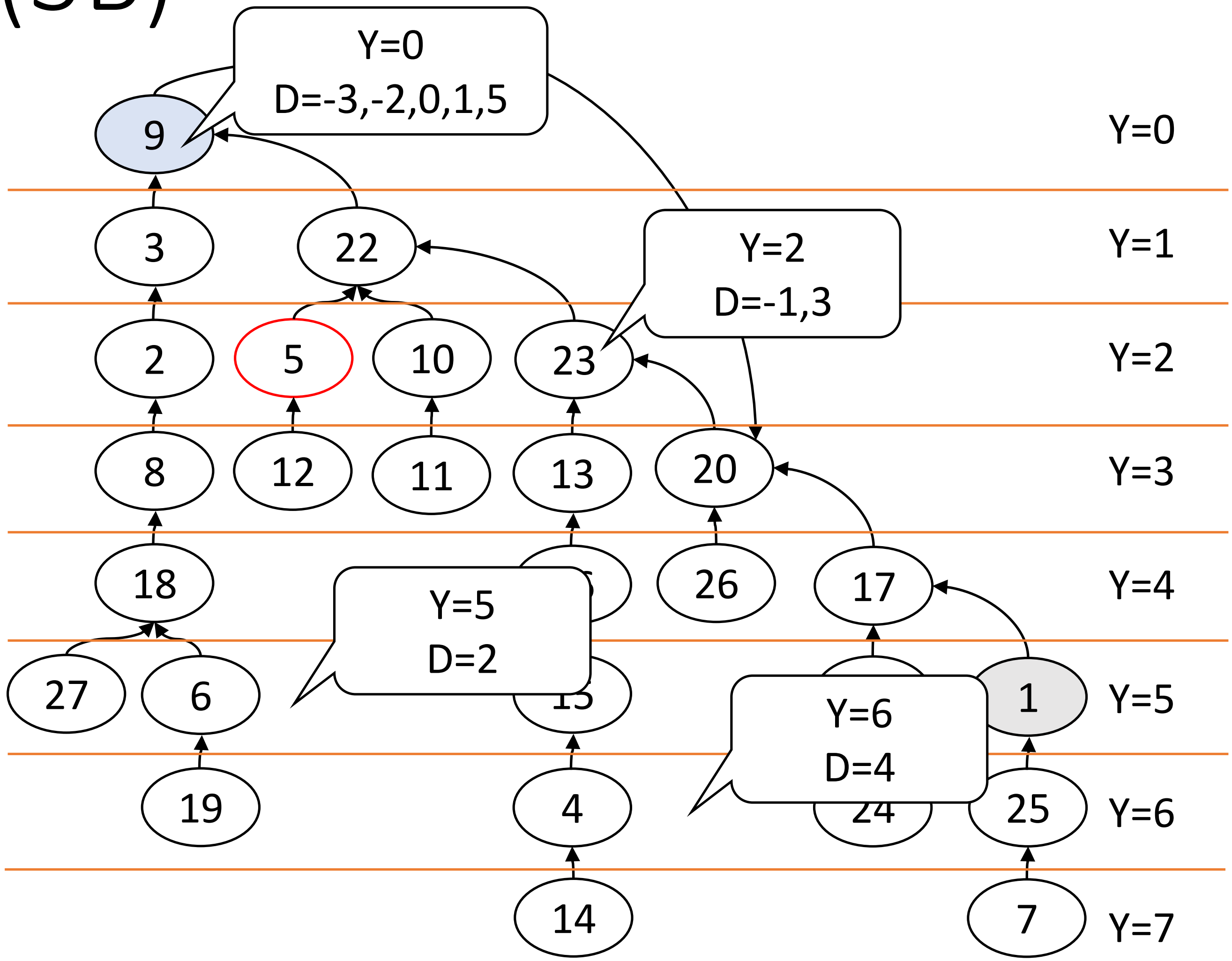
満点解法 – 場合分け(3B)

- 実際の数え上げ:
  下から走査する
- このときループで回す頂点は
  頂点Aの候補ではなく、
  到達先の頂点の候補を
  意味する



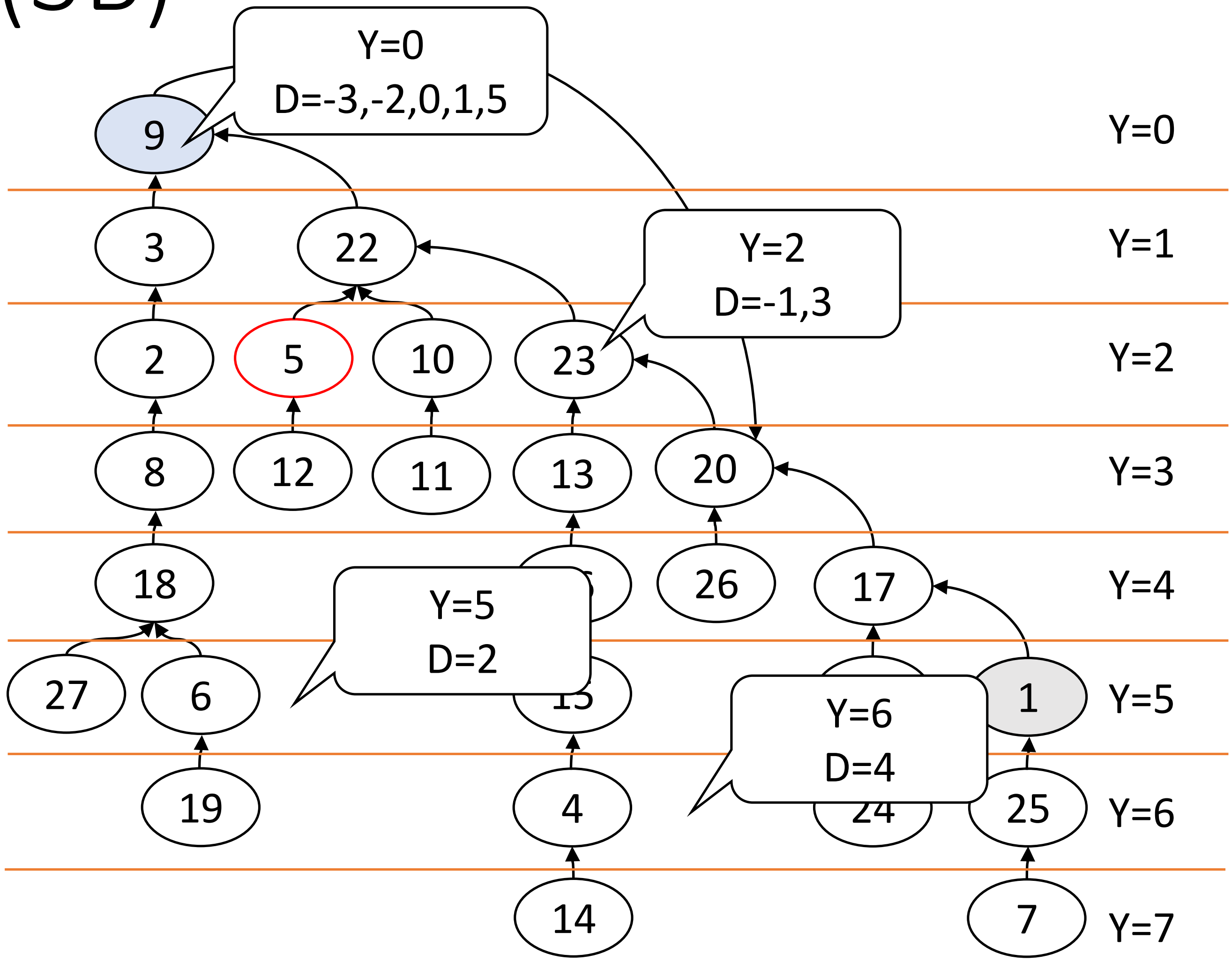
満点解法 – 場合分け(3B)

- 各到達先候補について:



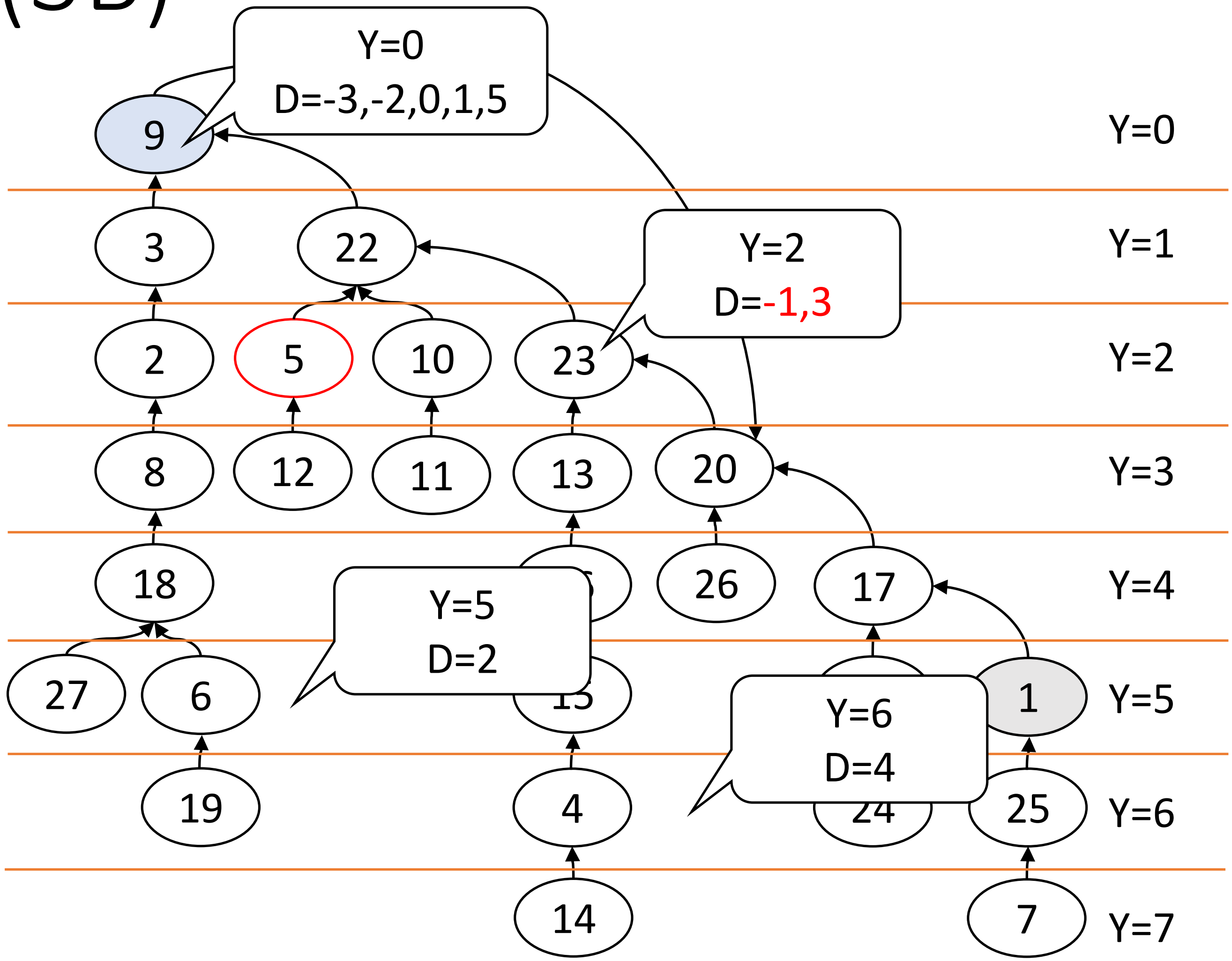
満点解法 – 場合分け(3B)

- 各到達先候補について:
  - Dの候補を列挙する



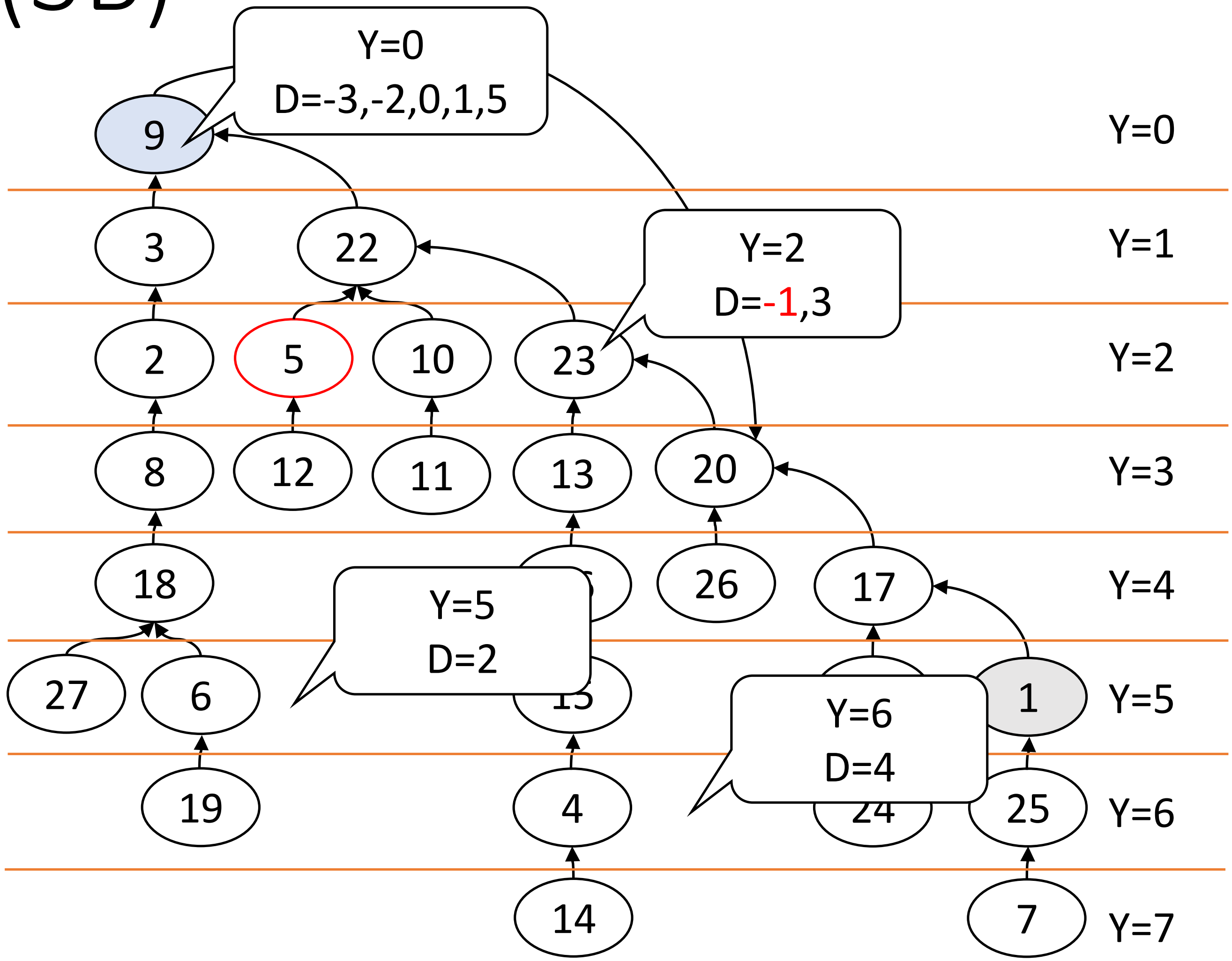
満点解法 – 場合分け(3B)

- 各到達先候補について:
  - Dの候補を列挙する
    - この場合は-1, -3



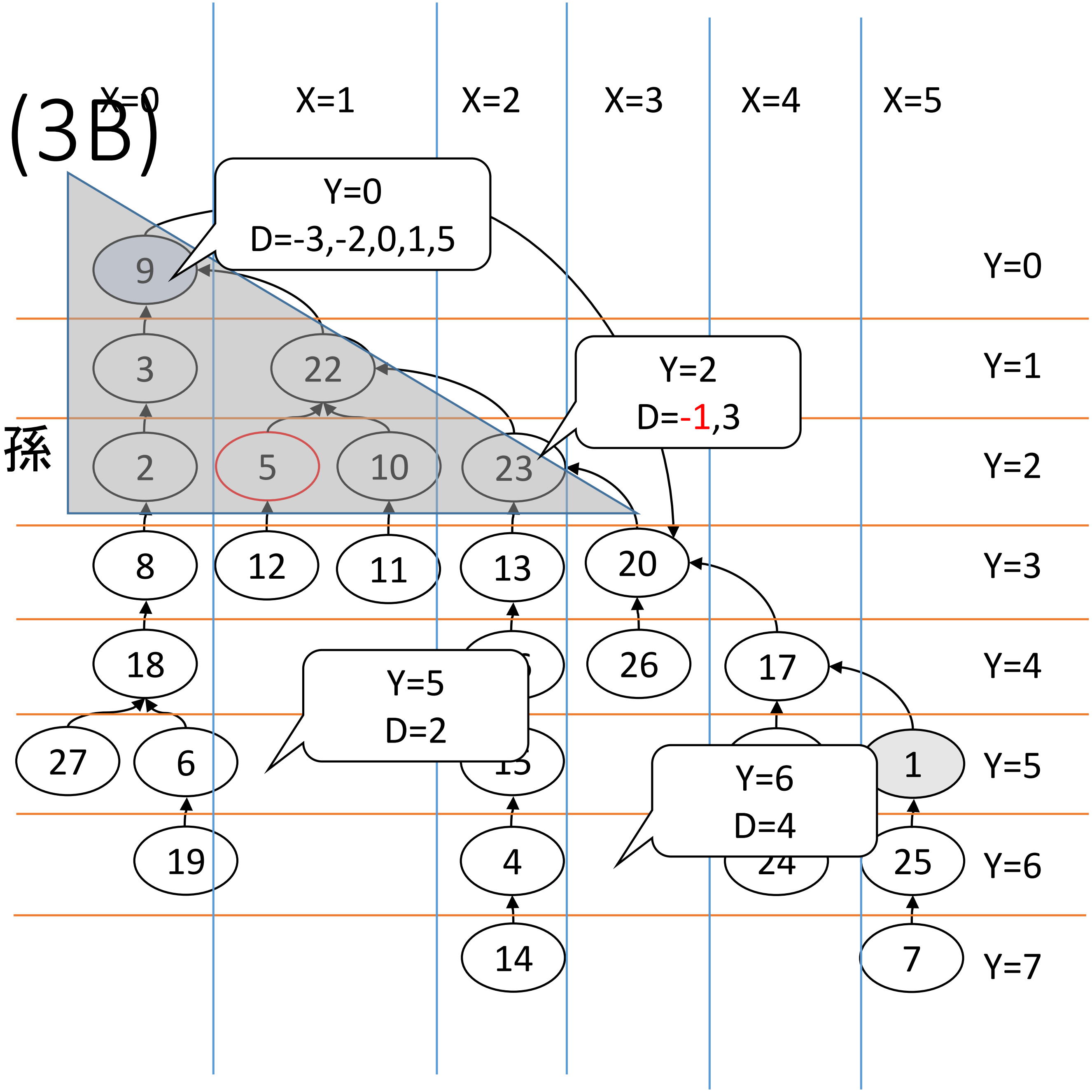
満点解法 – 場合分け(3B)

- 各(到達先候補,D)について:
  - Bの候補を列挙する



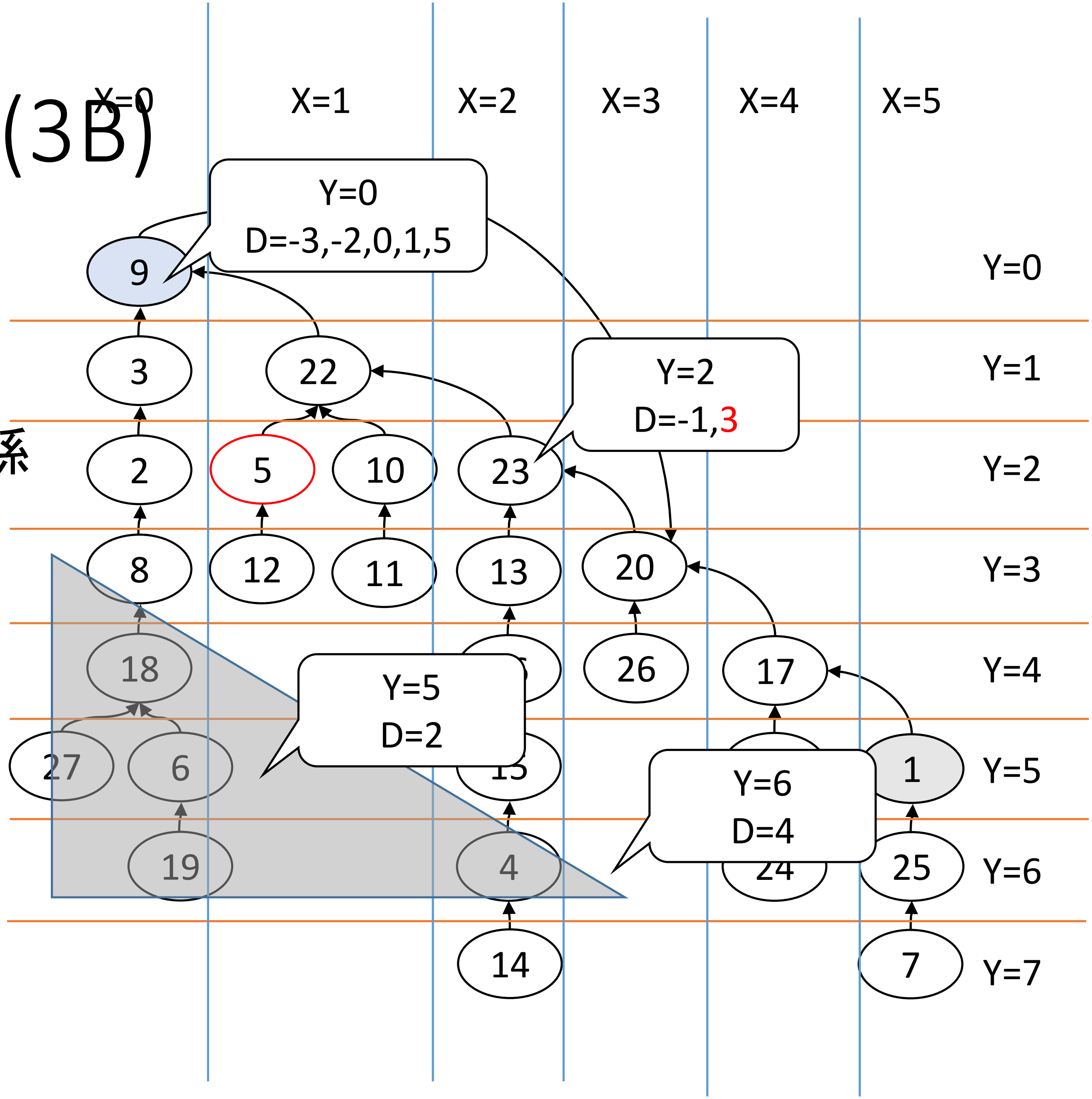
満点解法 – 場合分け(3B)

- 各(到達先候補,D)について:
  - Bの候補を列挙する
  - D=-1のとき、候補はこの範囲の子孫
    - この三角形はDのみに依存



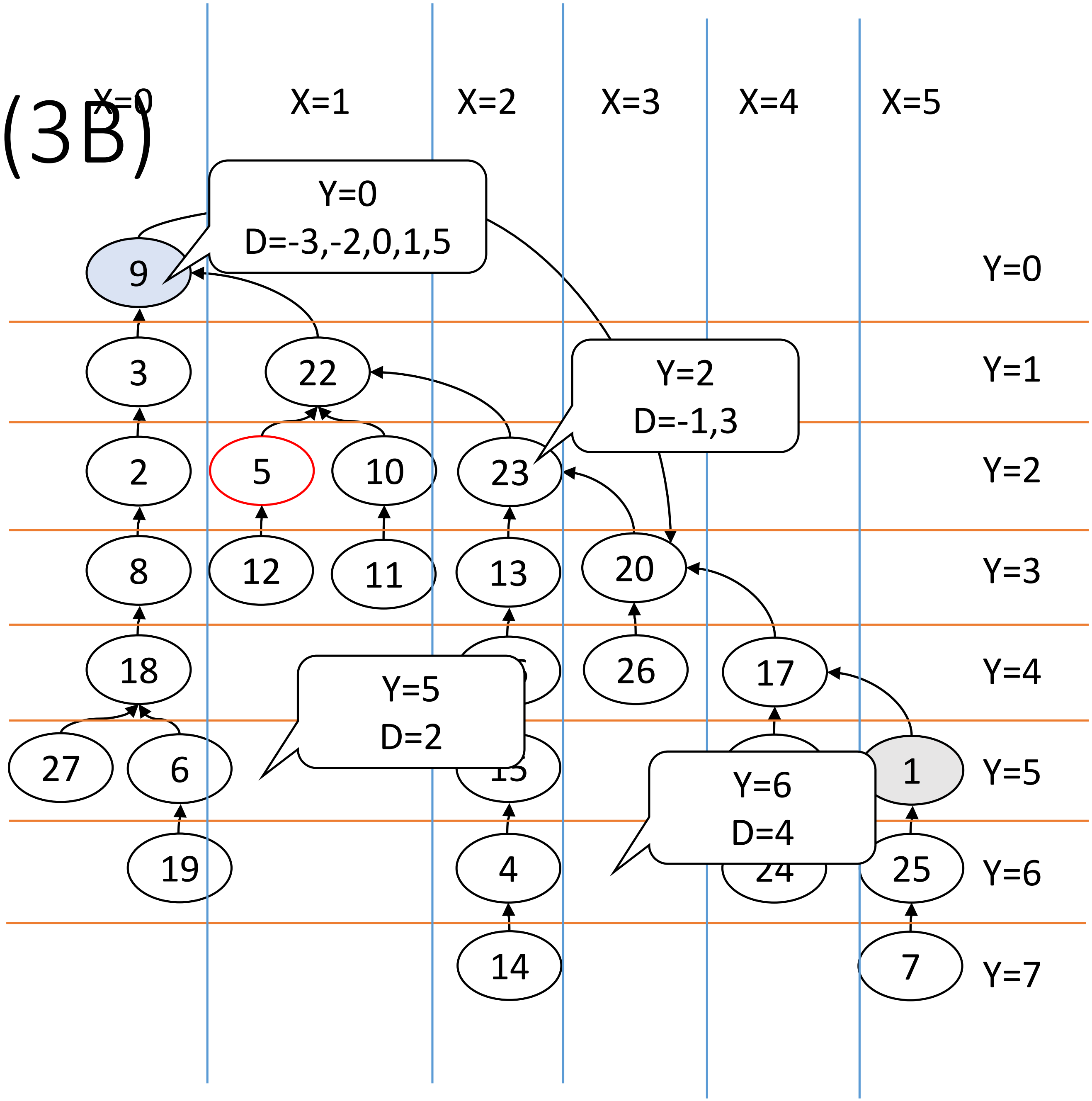
満点解法 – 場合分け(3B)

- 各(到達先候補,D)について:
  - Bの候補を列挙する
  - D=3のとき、候補はこの範囲の子孫
    - この三角形はDのみに依存



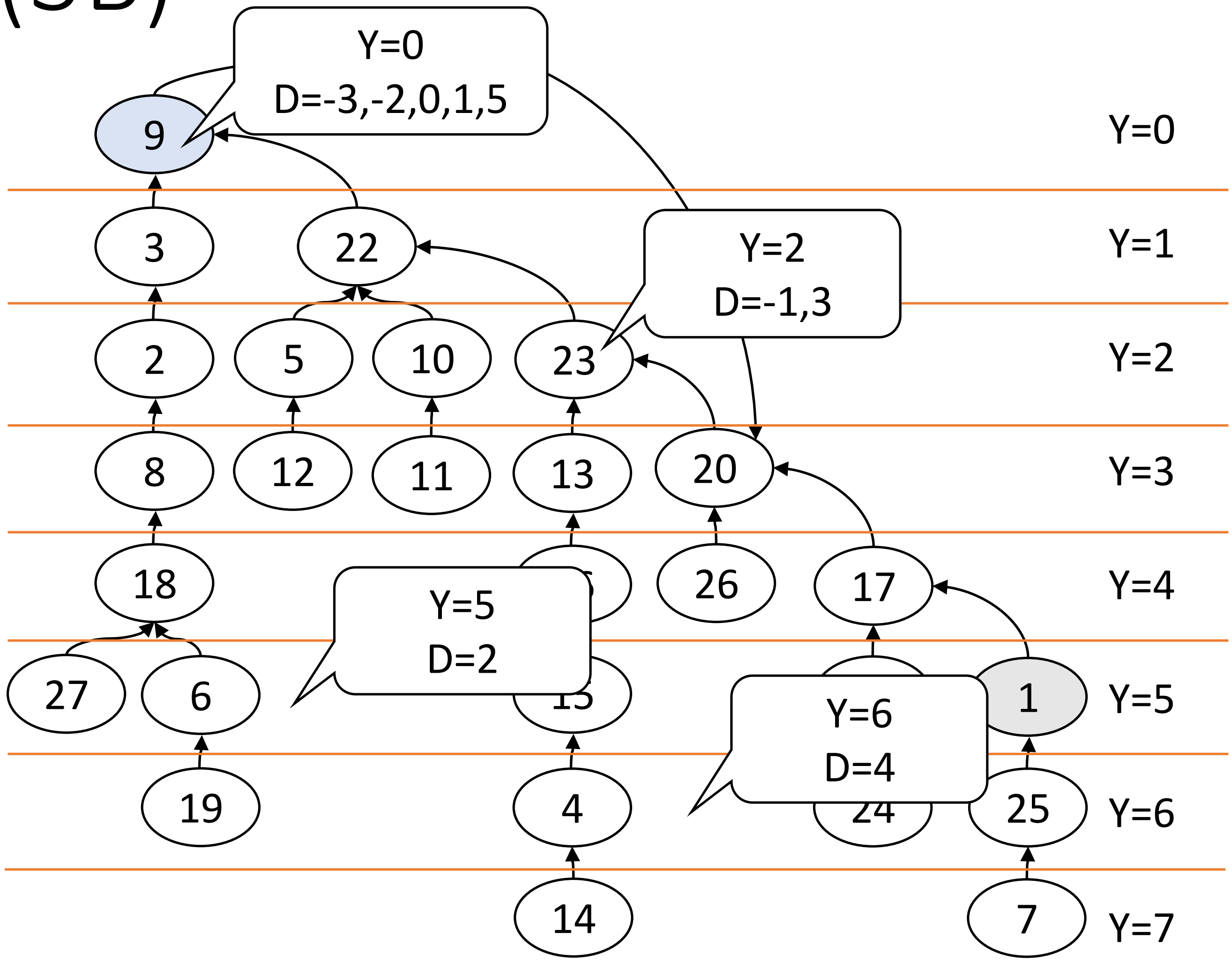
満点解法 – 場合分け(3B)

- 各(到達先候補,D)について:
  - Bの候補を列挙する
  - Bに対してAは一意に定まる
    - Bの候補の個数を数えるだけでよい



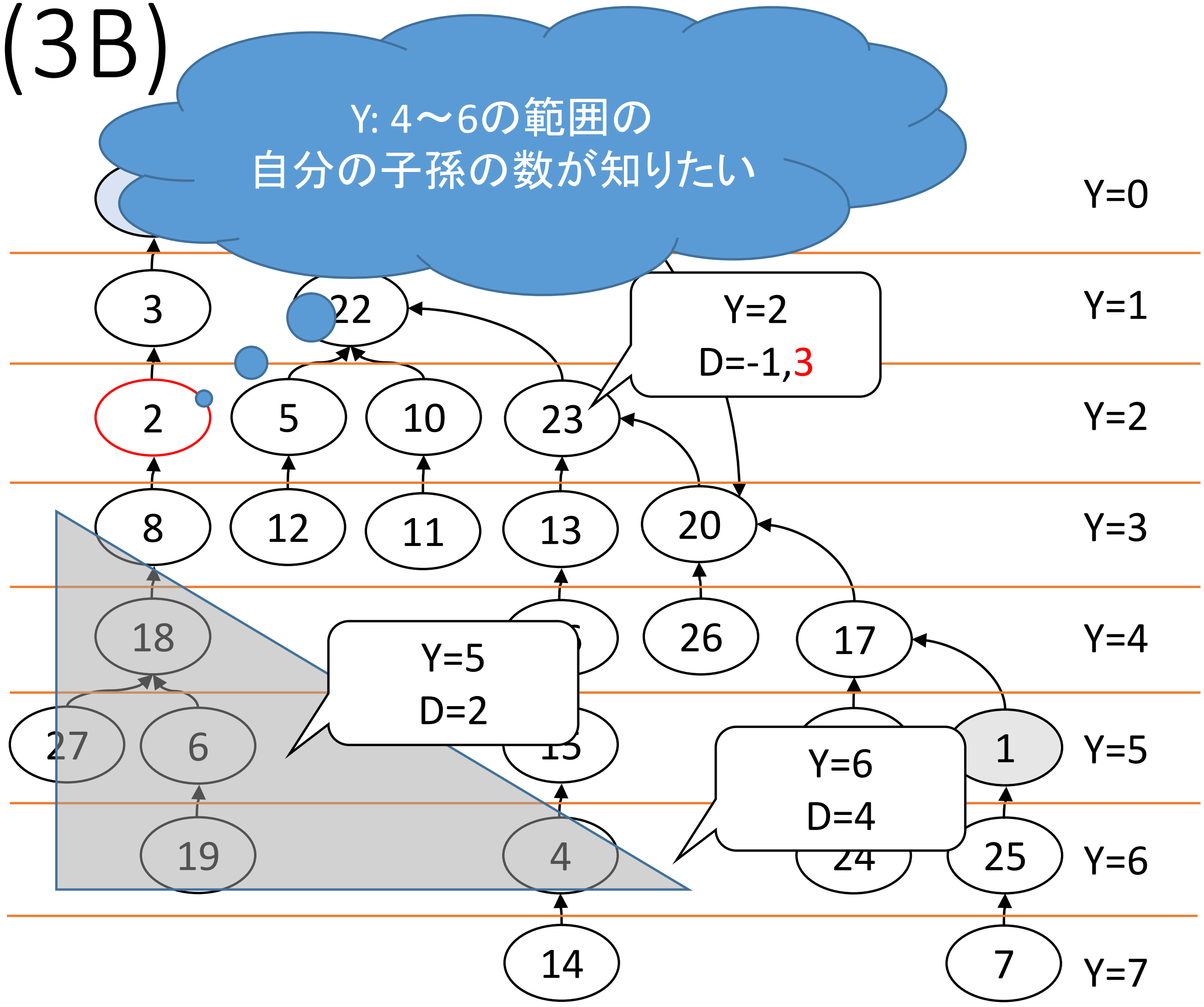
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい



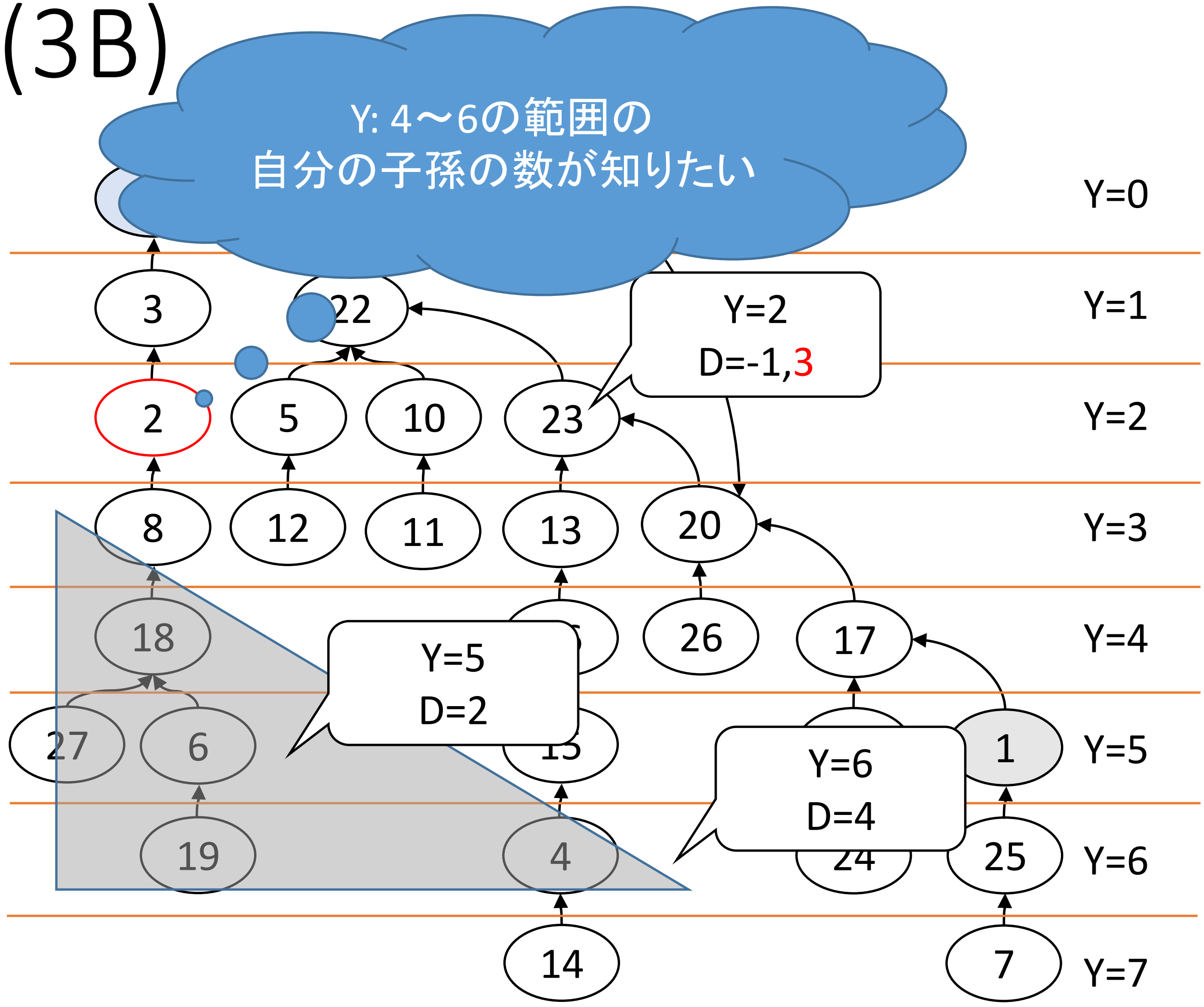
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい



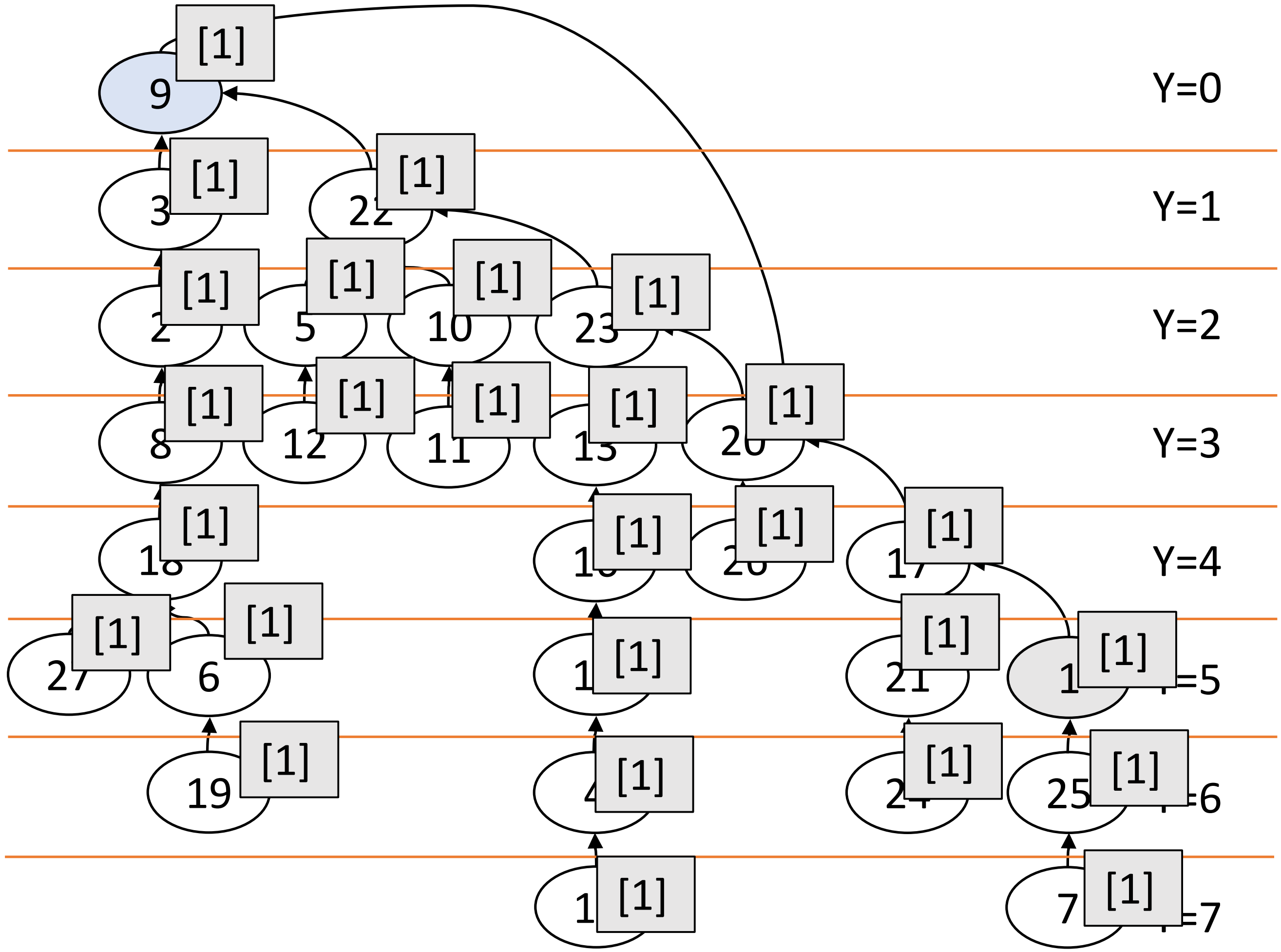
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
  - この場合は4個



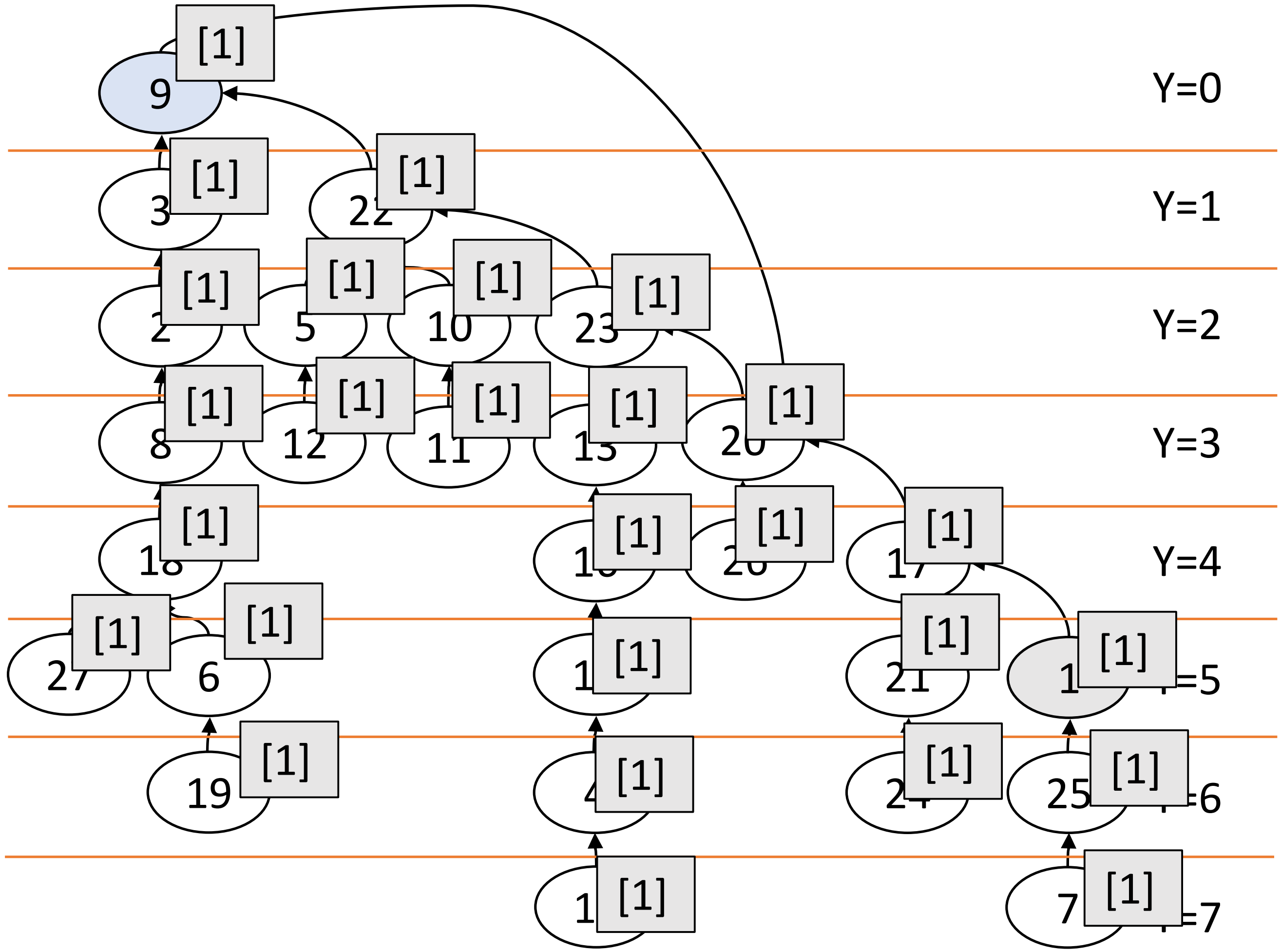
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化



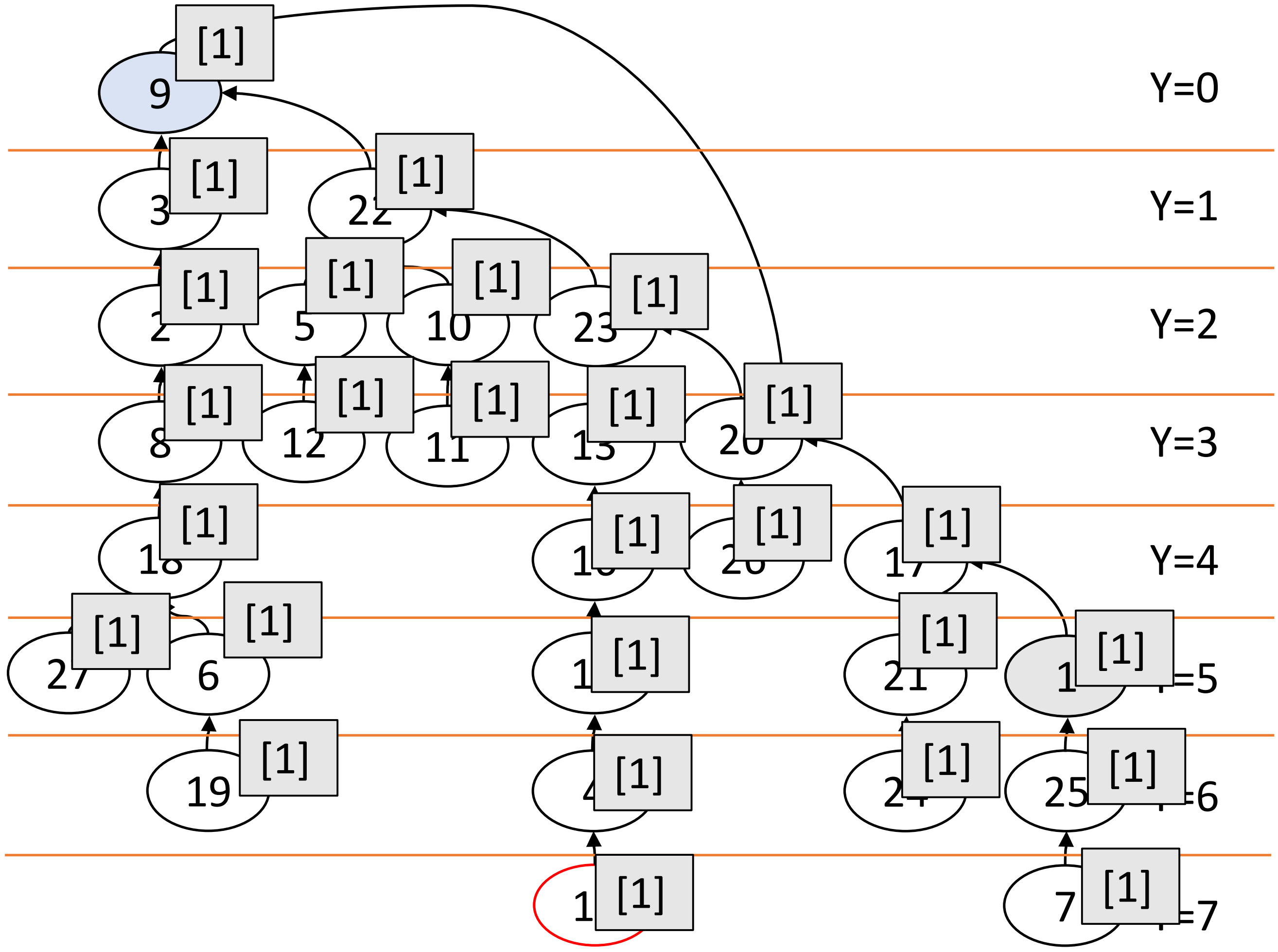
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



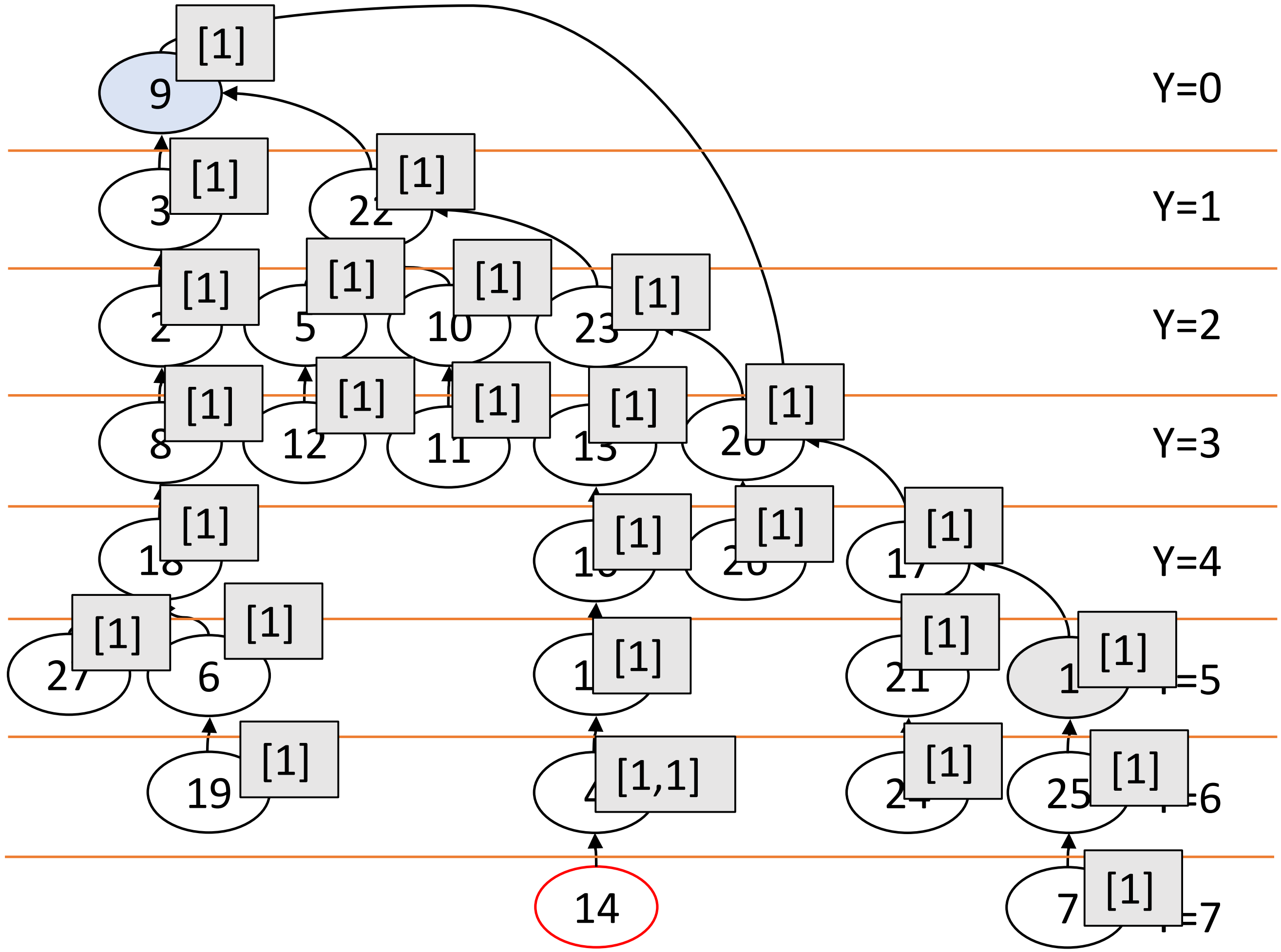
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査
  - 処理中の頂点のFenwick Treeは子孫の個数を正しく表している



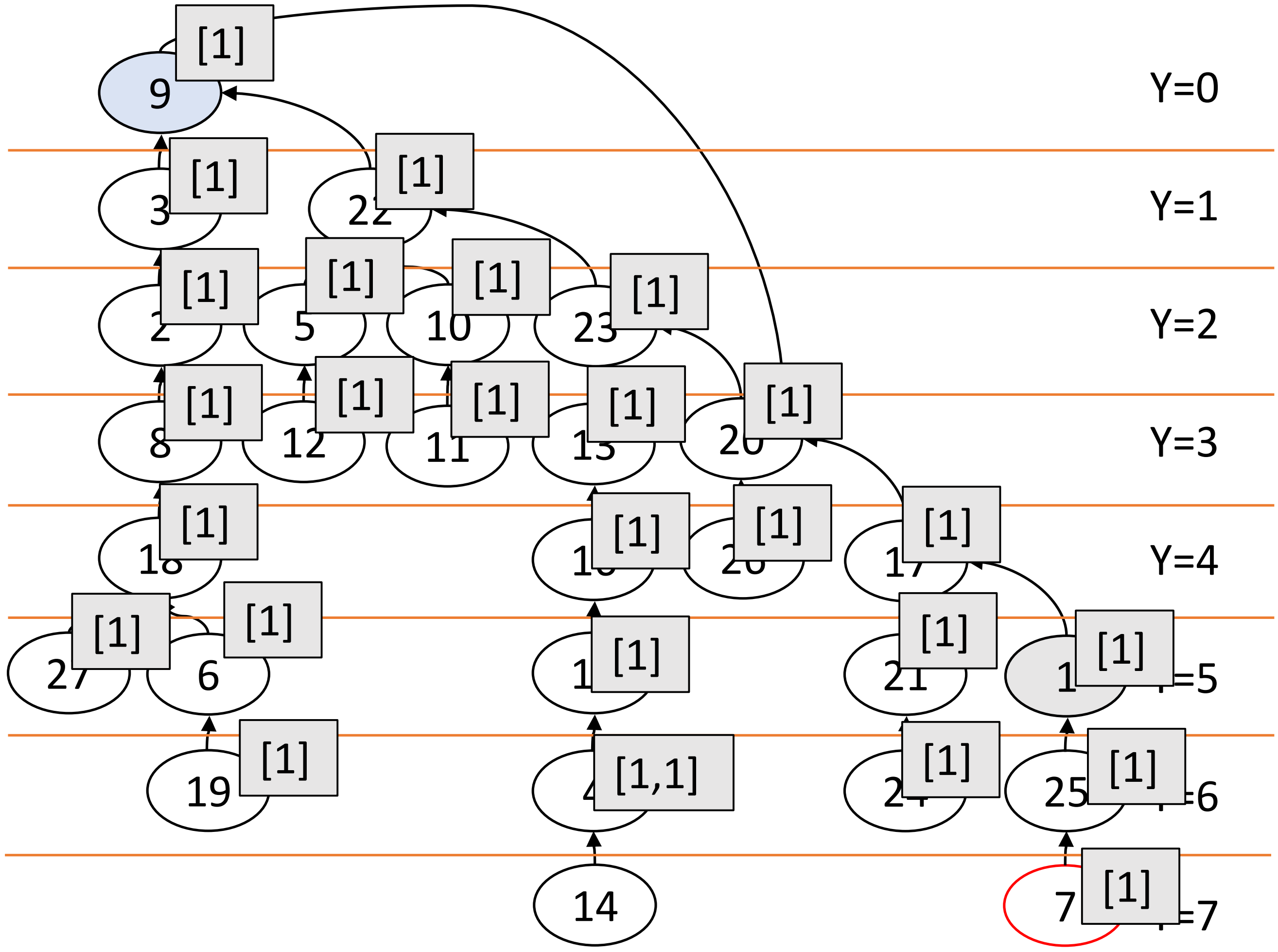
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査
  - 終わったら上の木とマージ



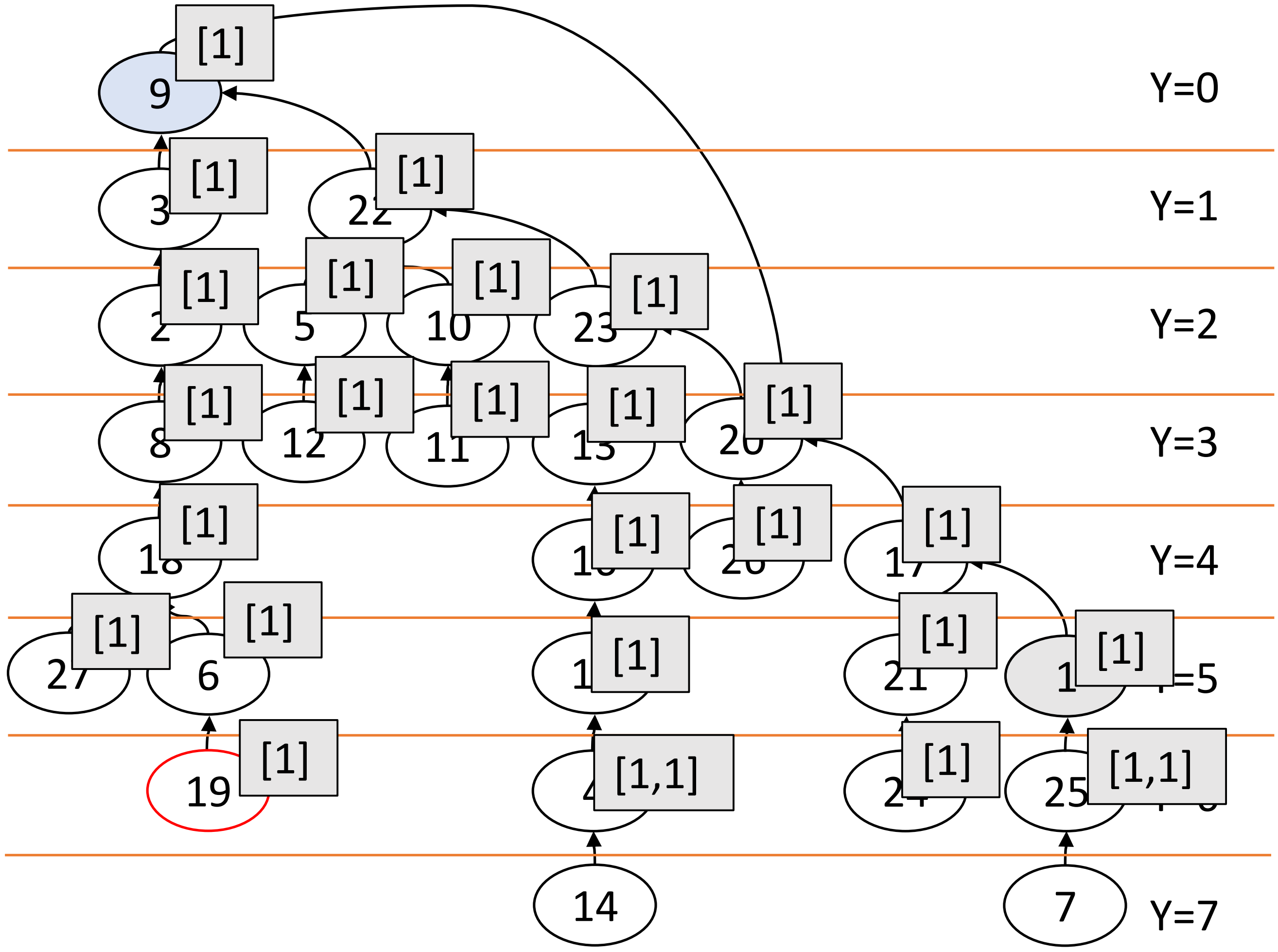
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



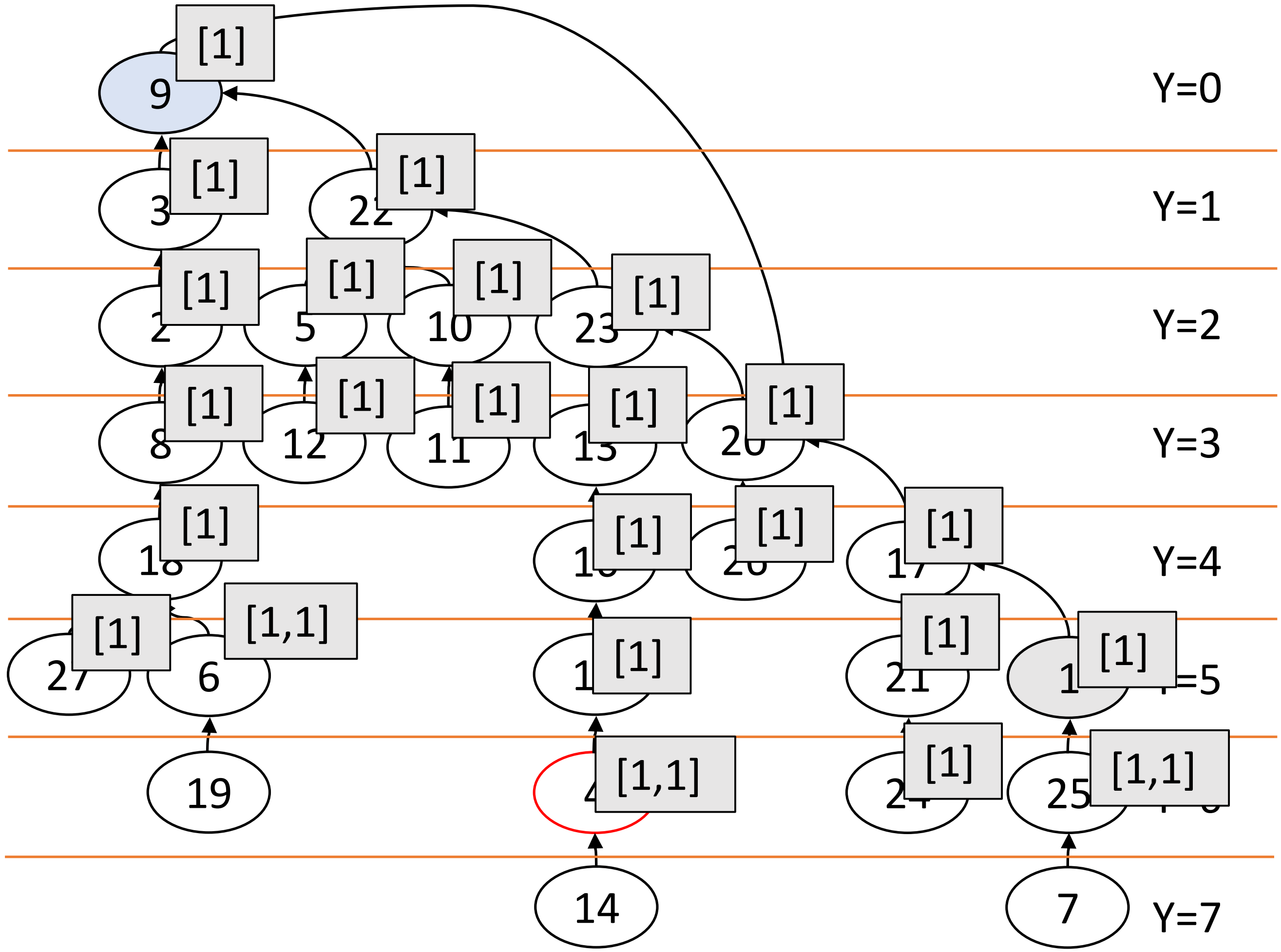
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



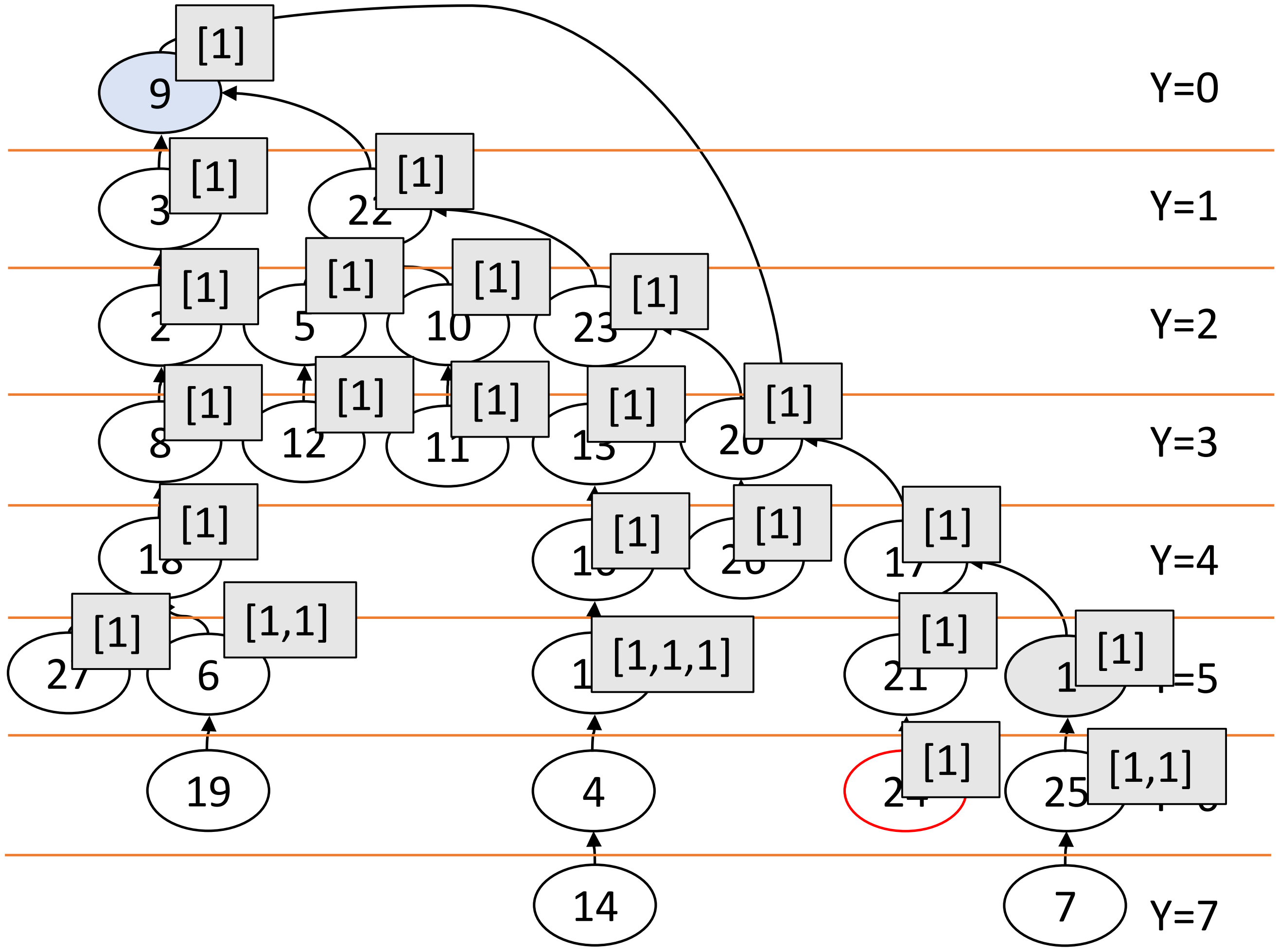
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



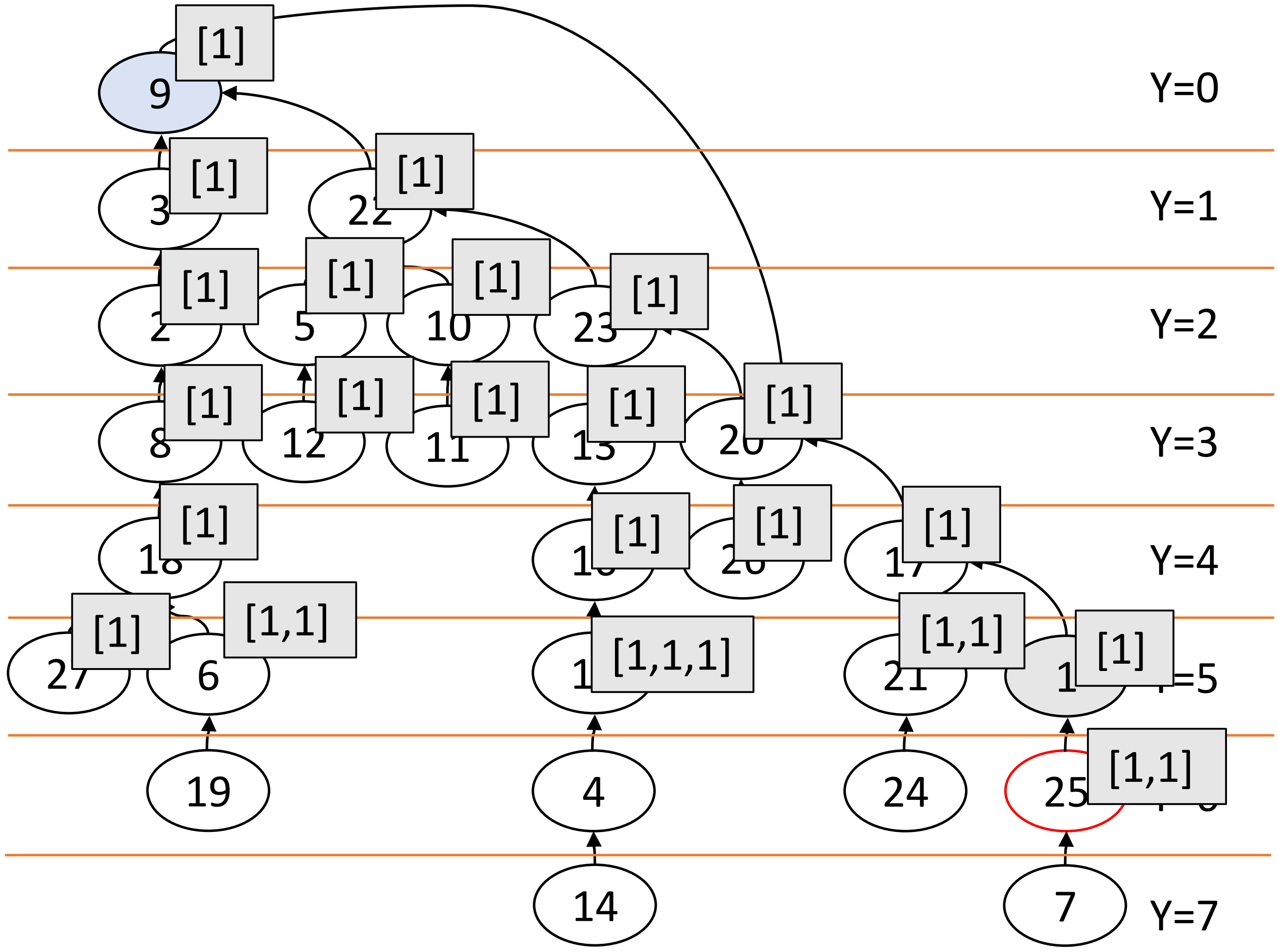
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



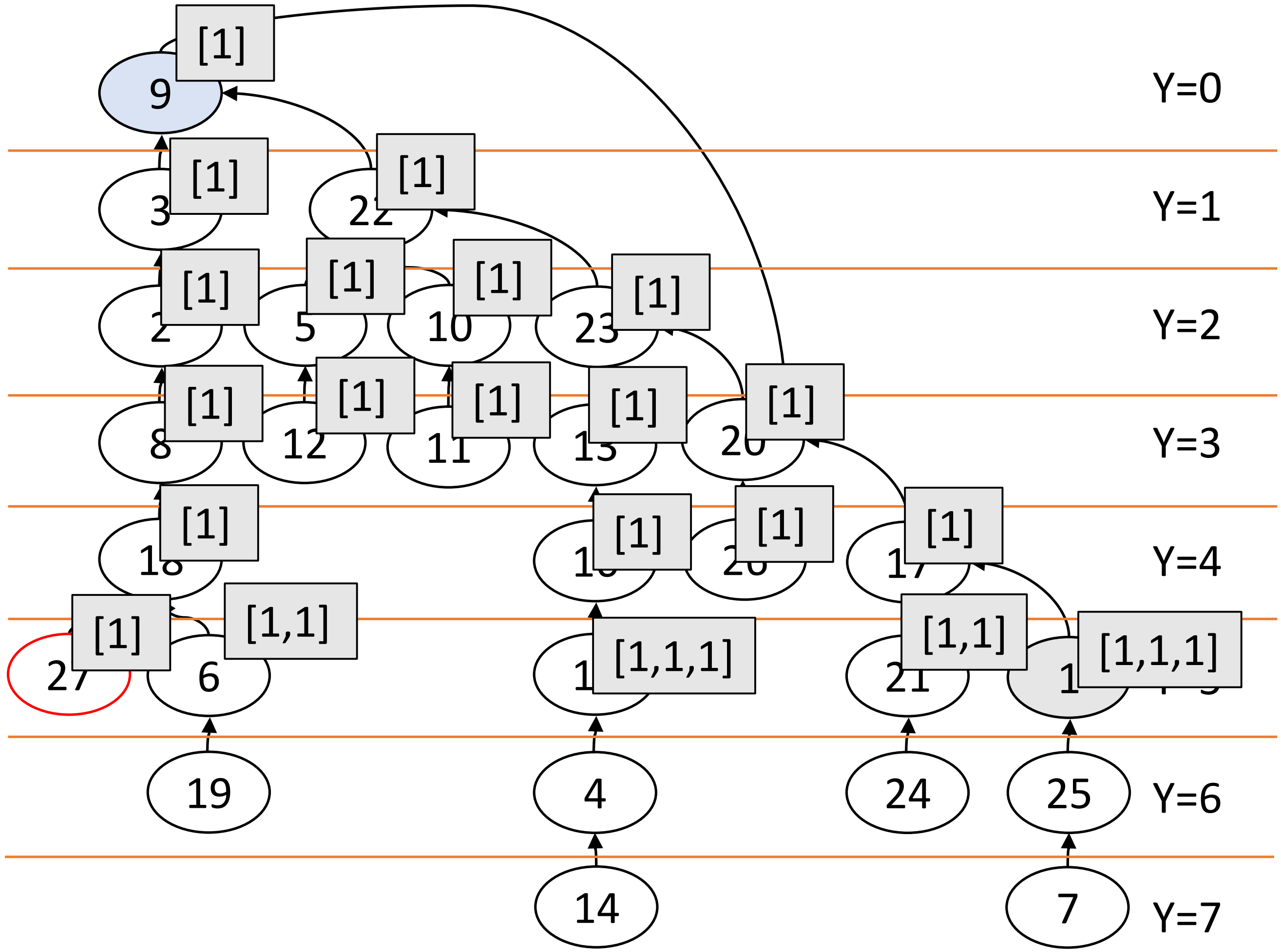
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



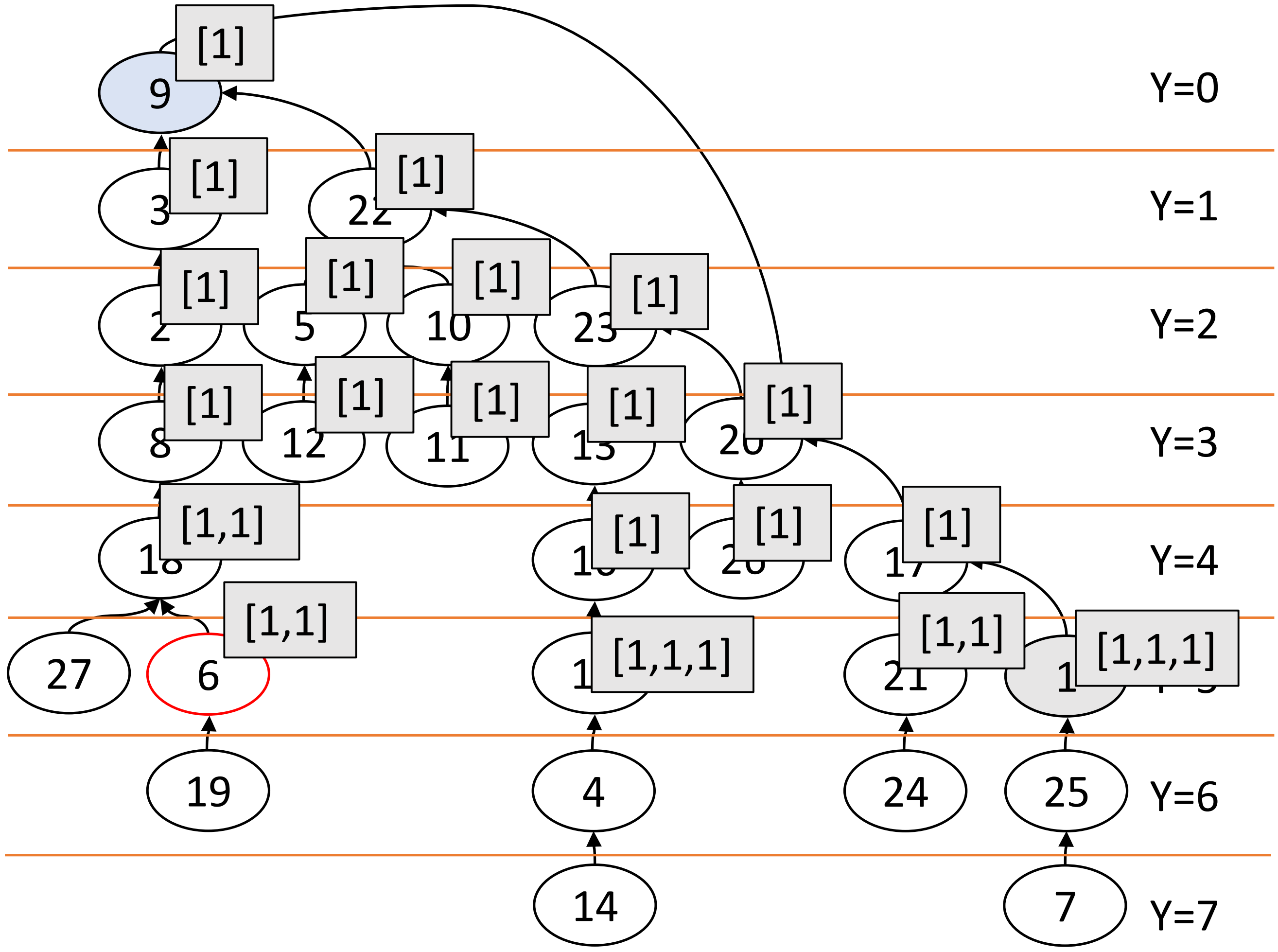
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



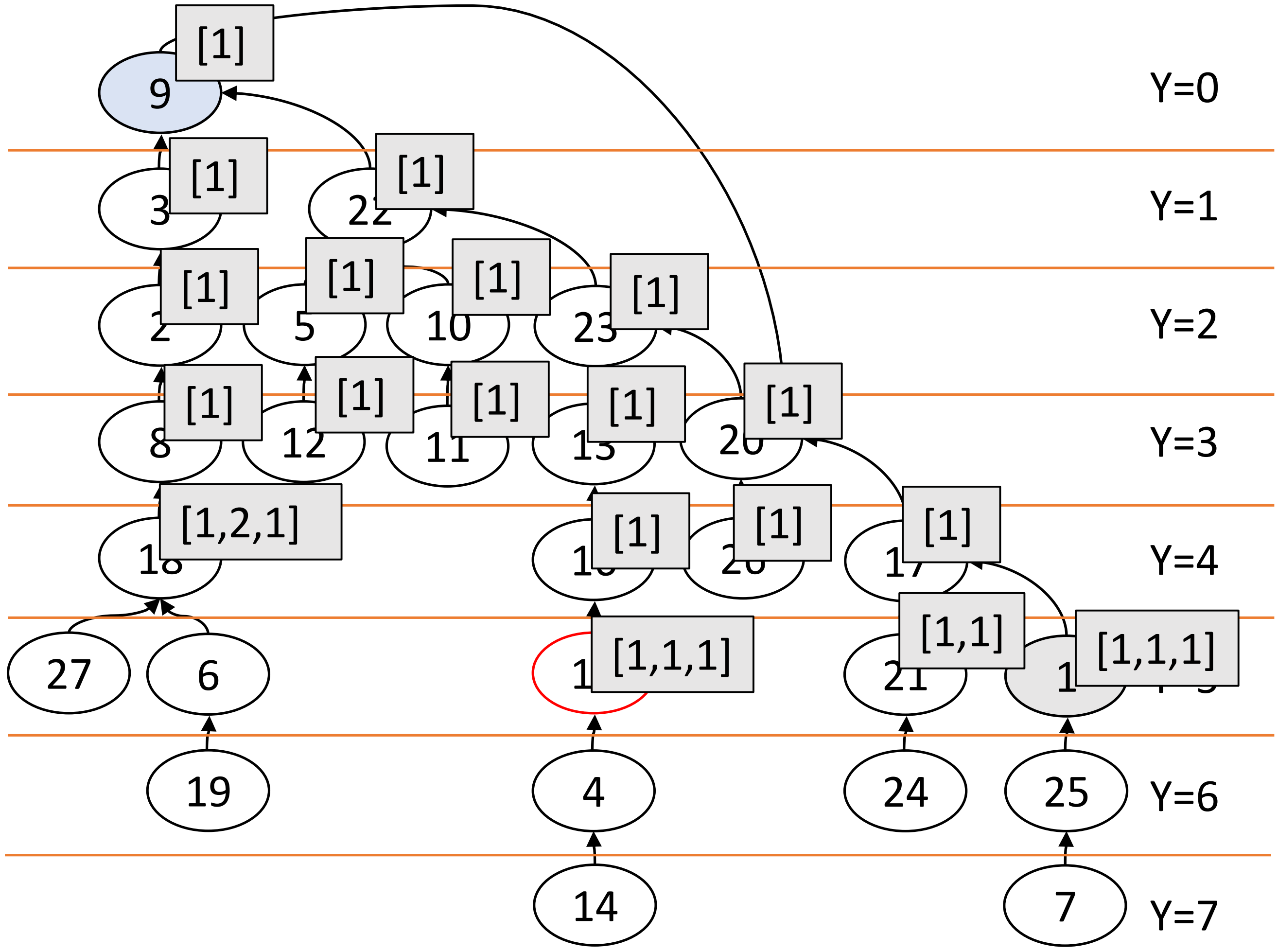
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



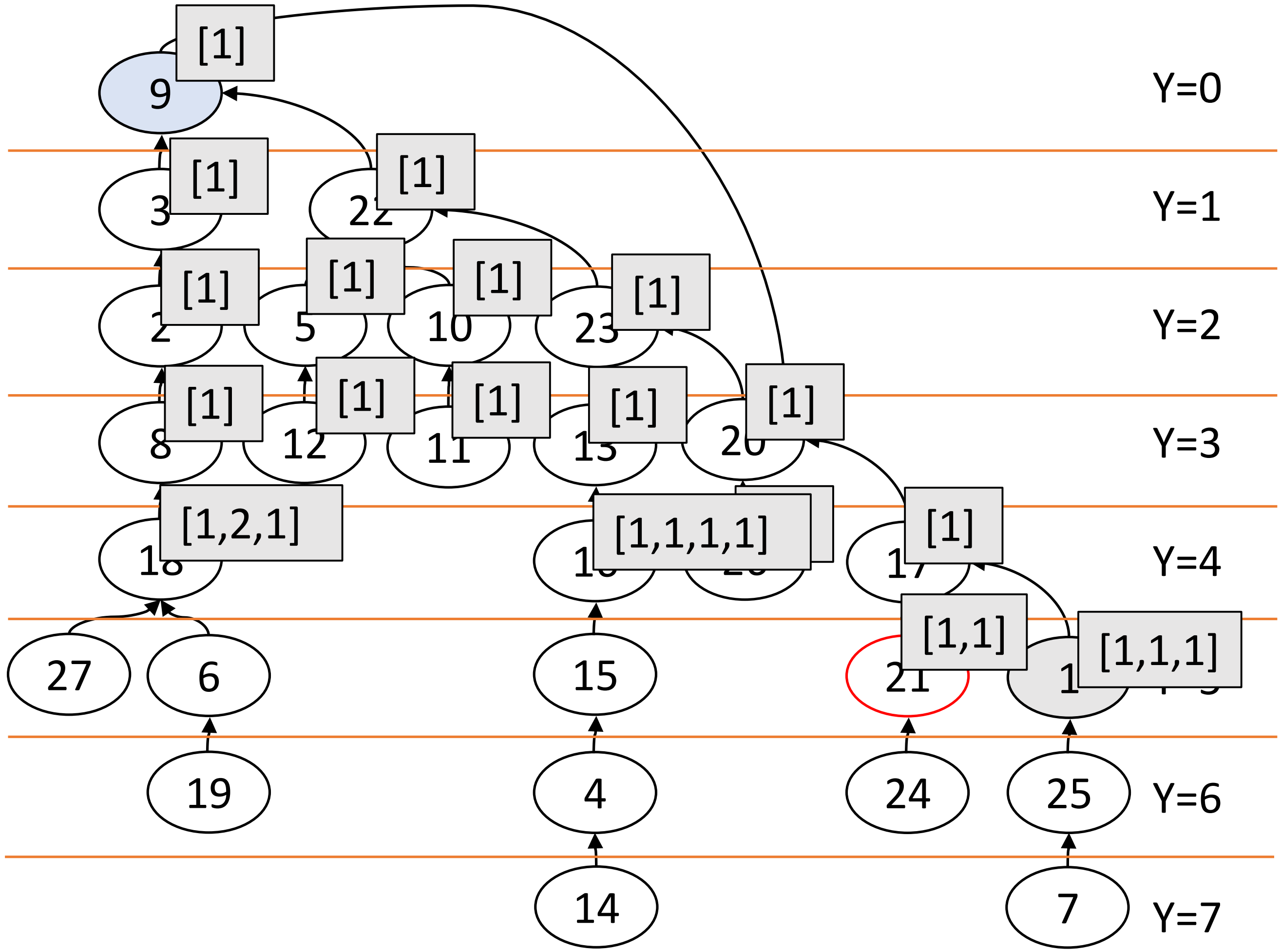
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



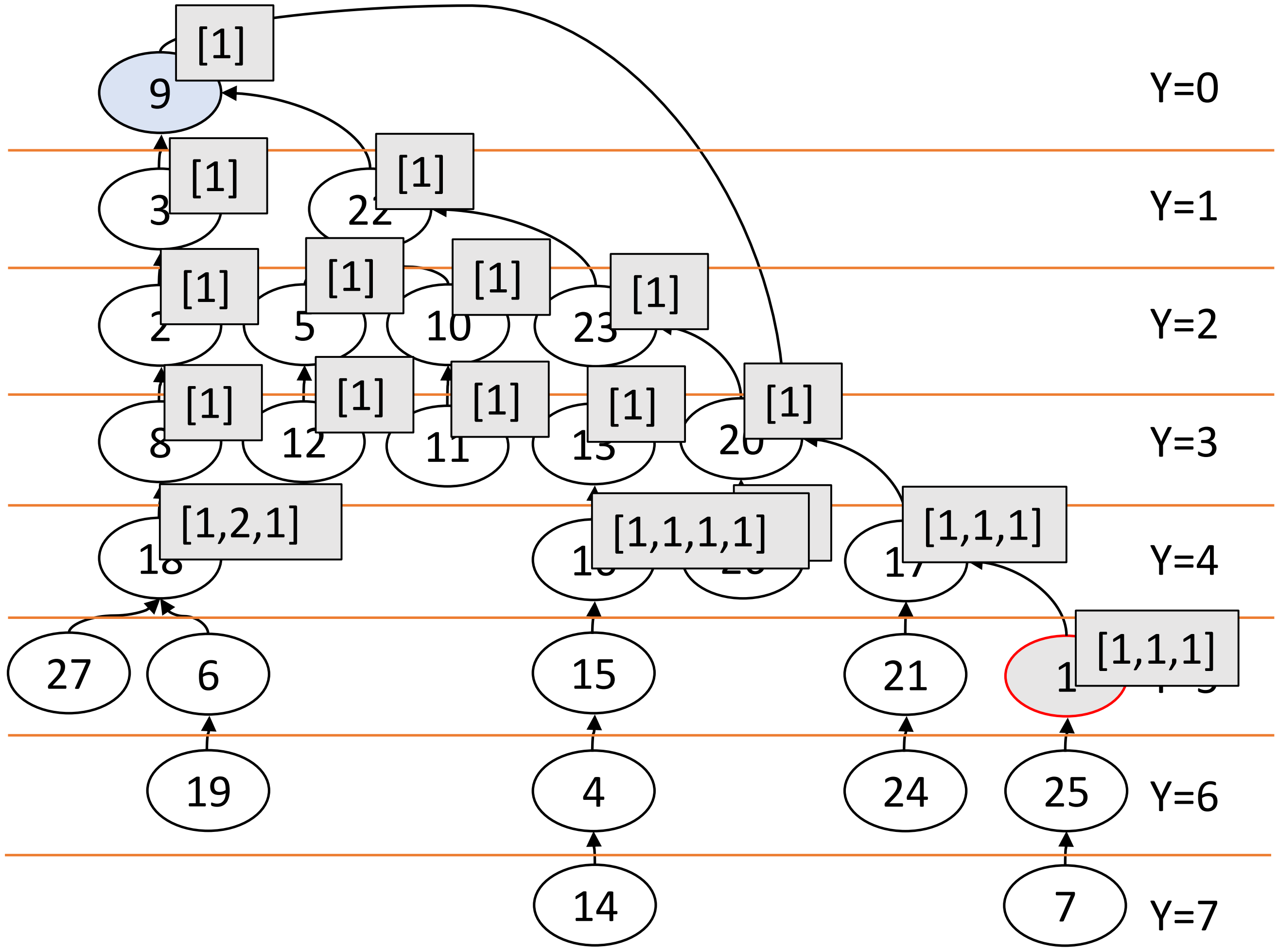
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



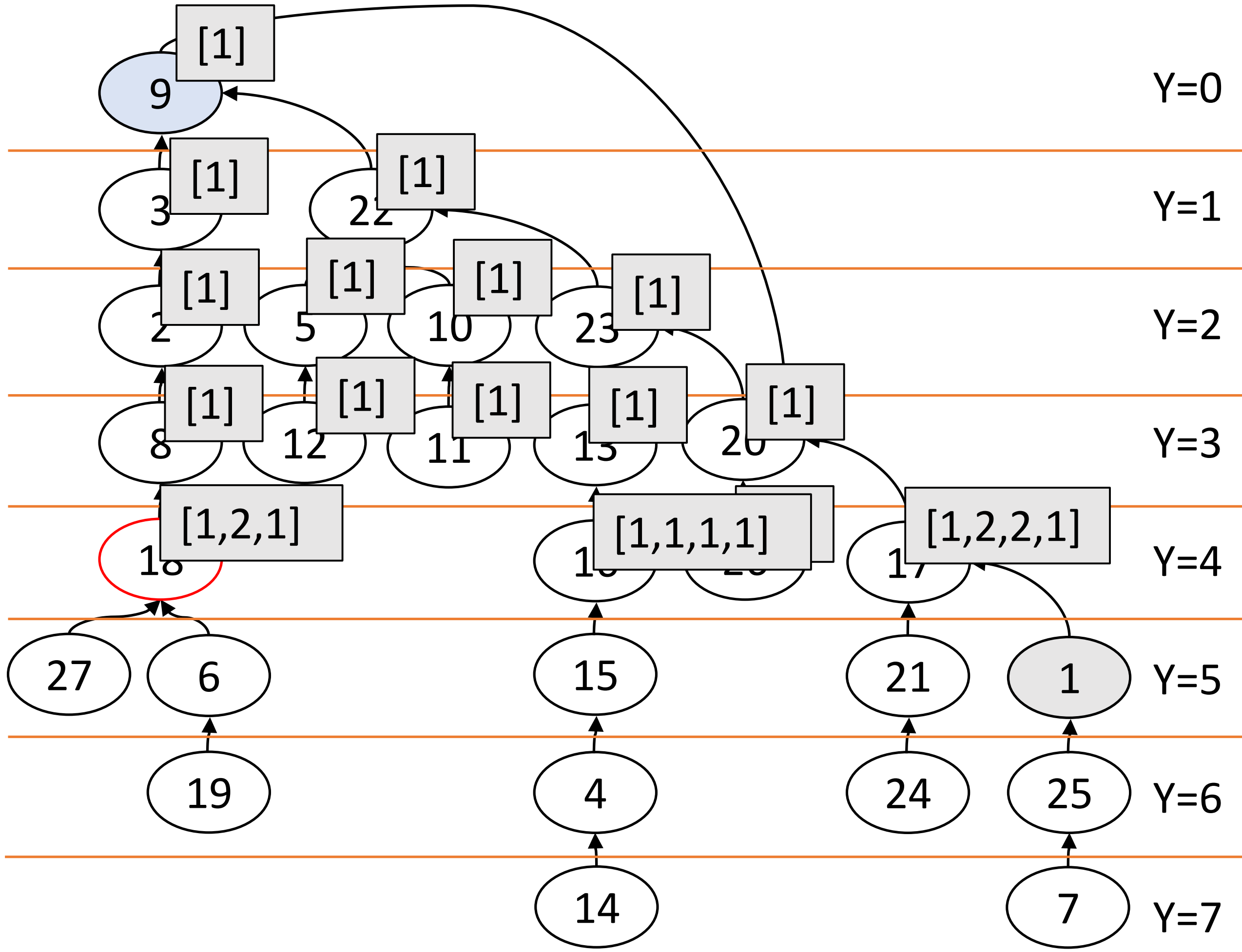
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



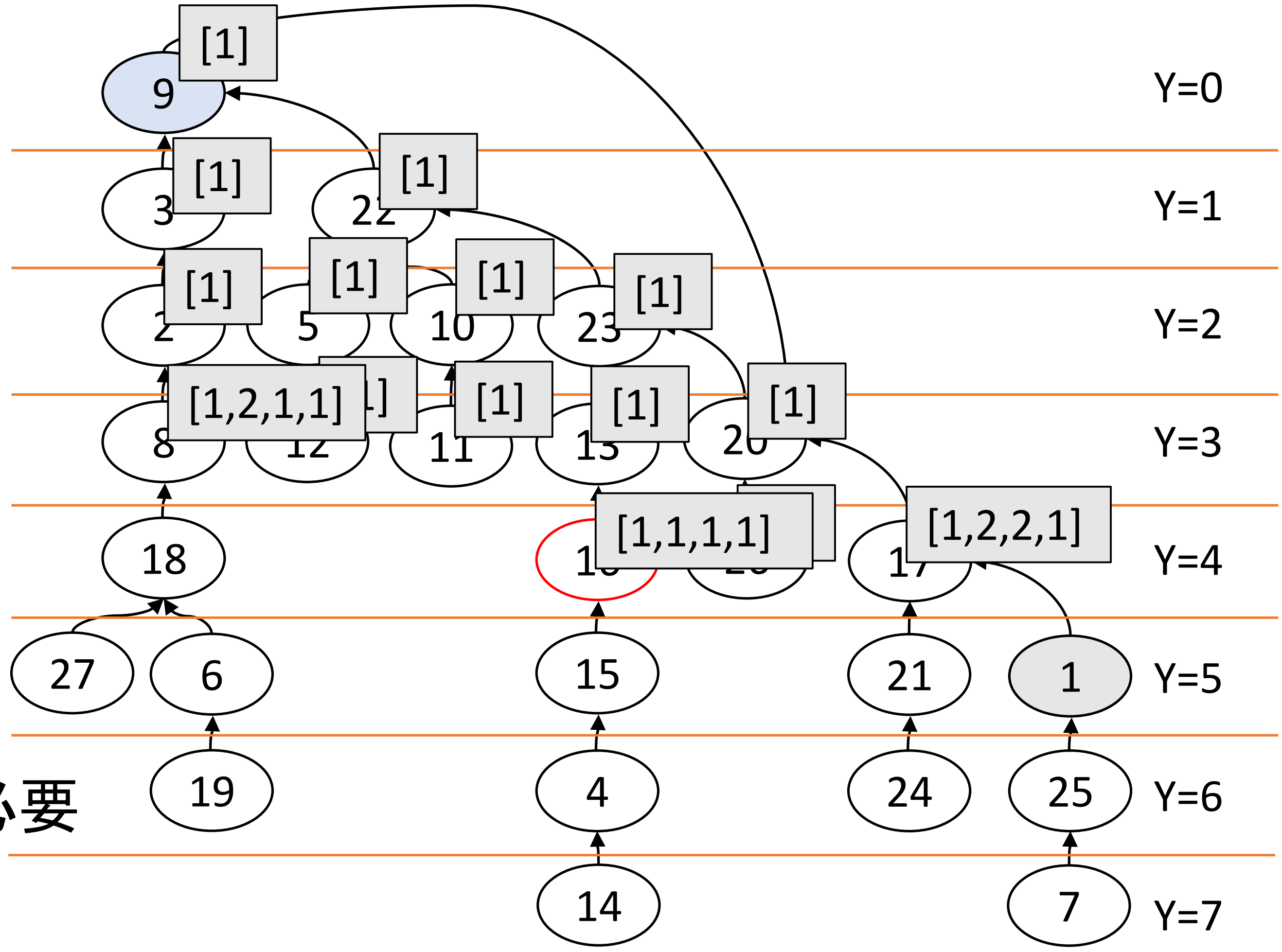
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



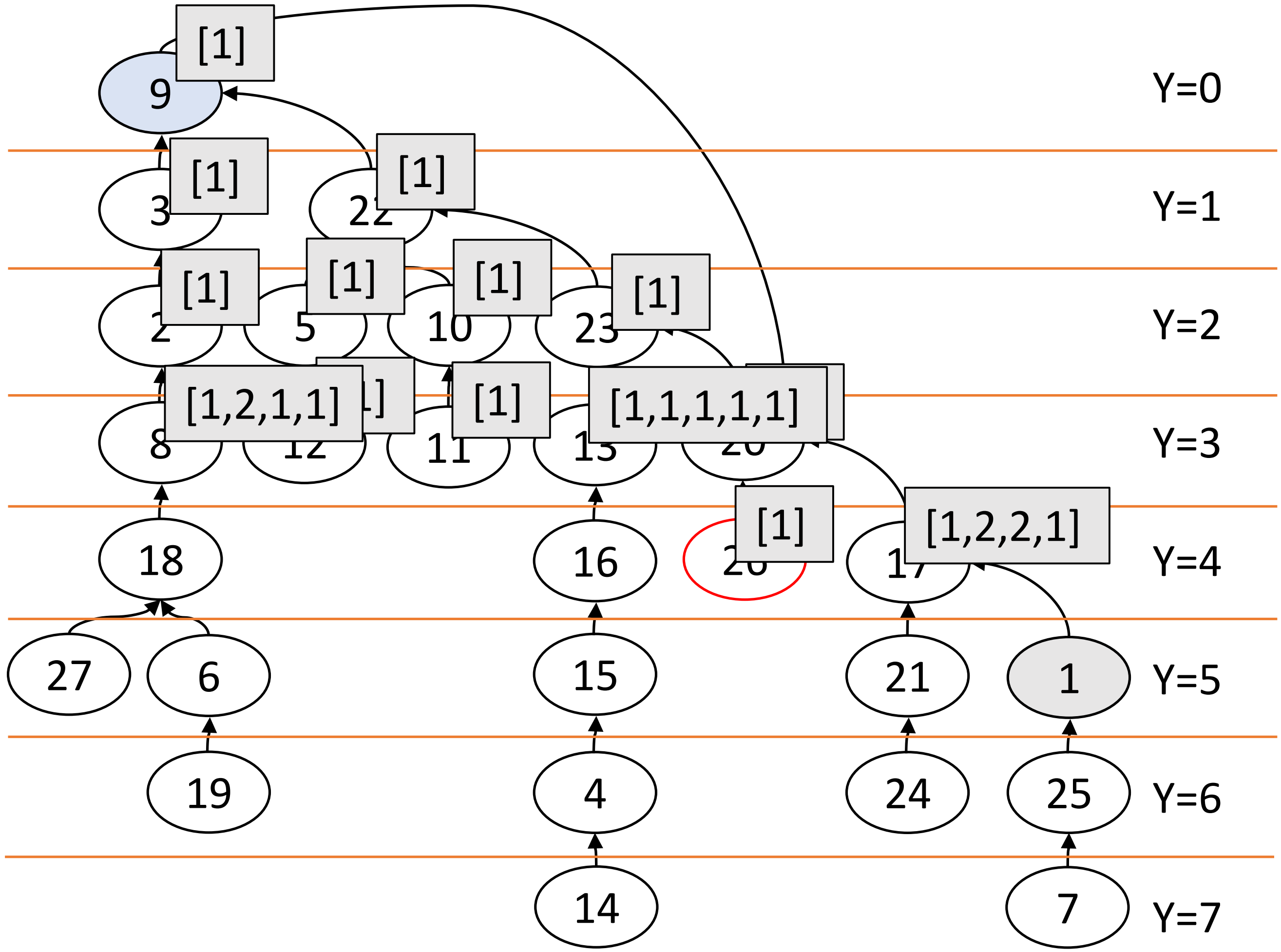
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査
  - Fenwick Treeの数字は、左ほど深い(Yが大きい)頂点の個数を表すように保持する
  - これはマージの効率化のために必要



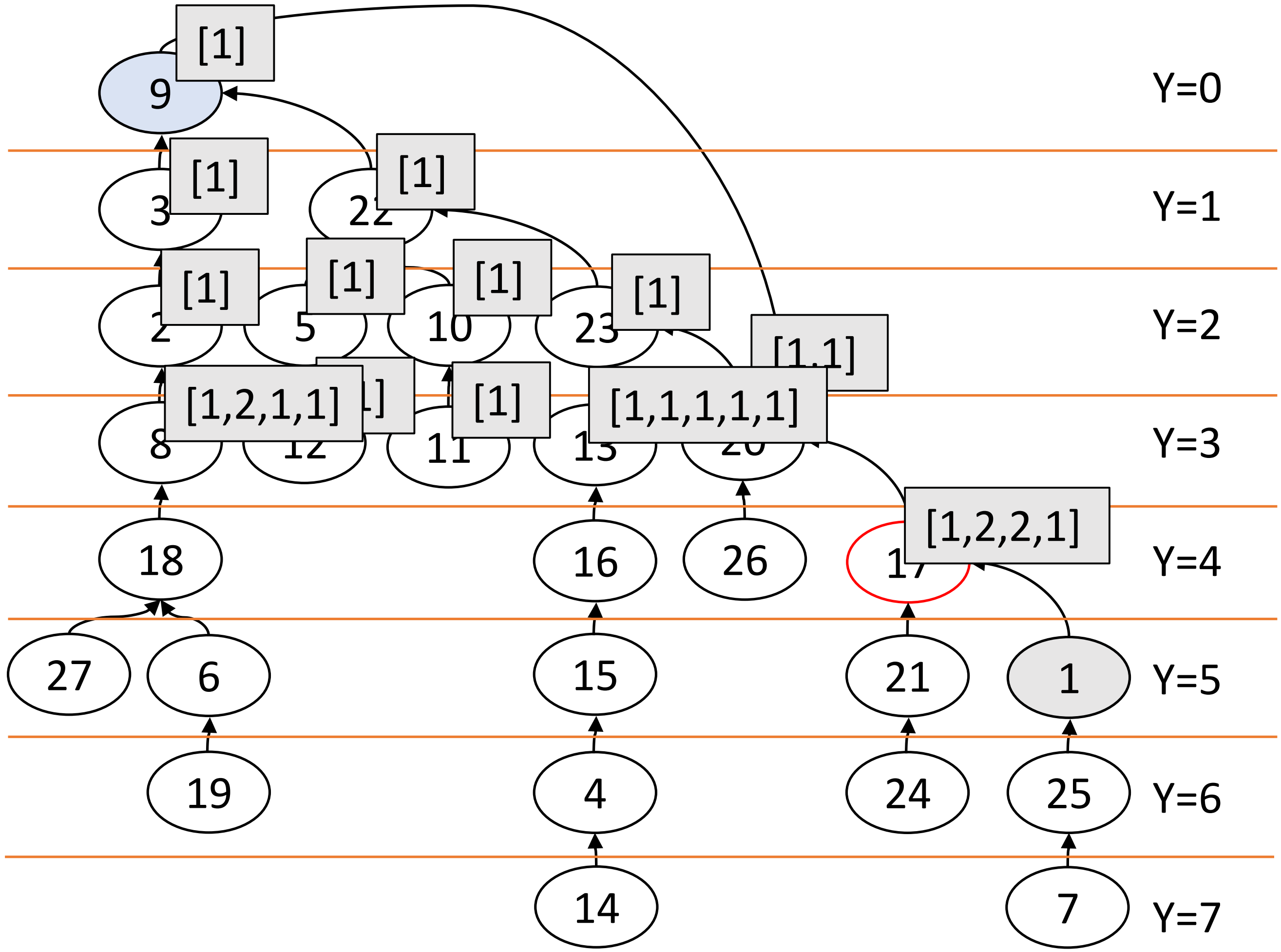
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



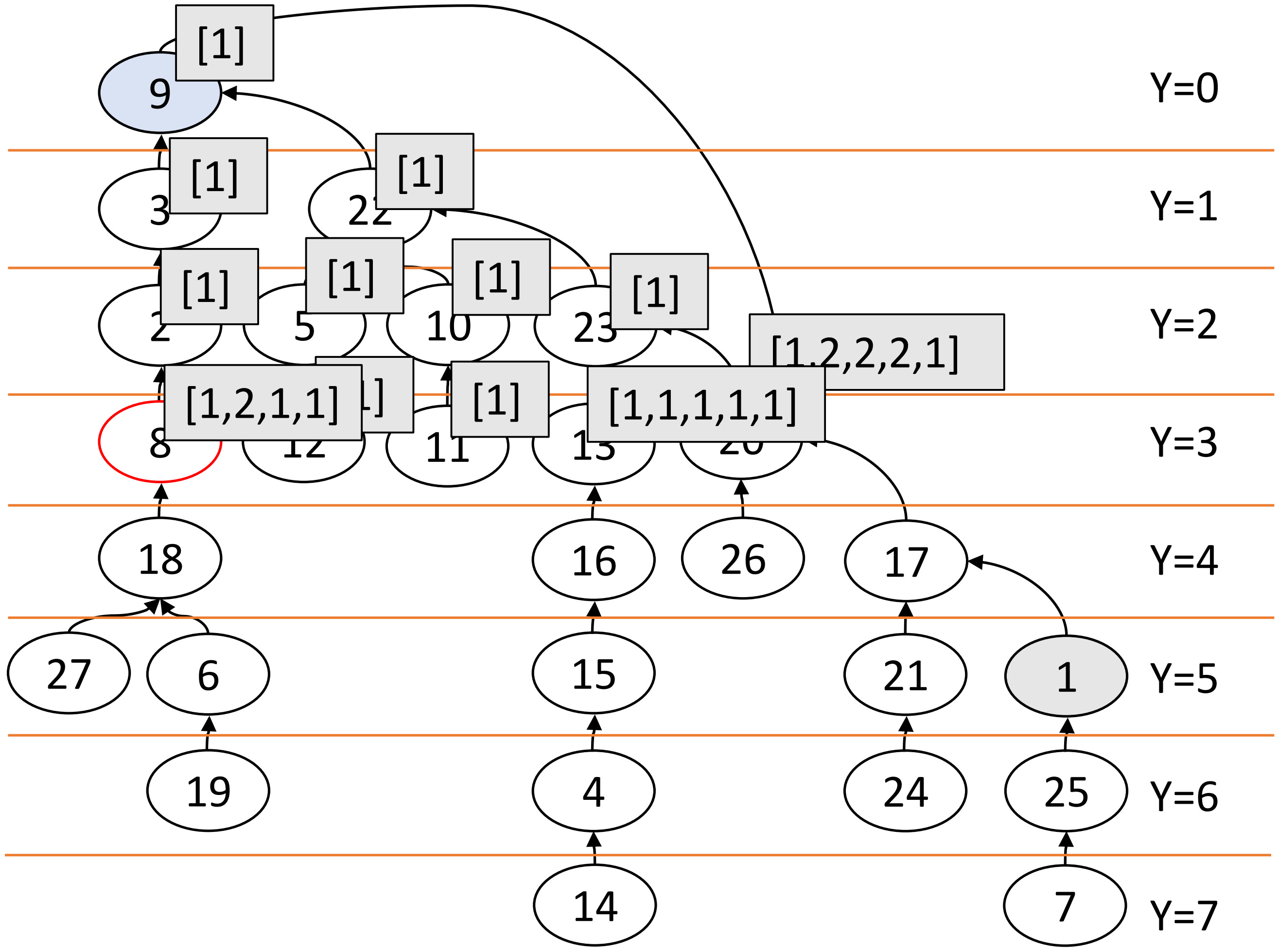
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



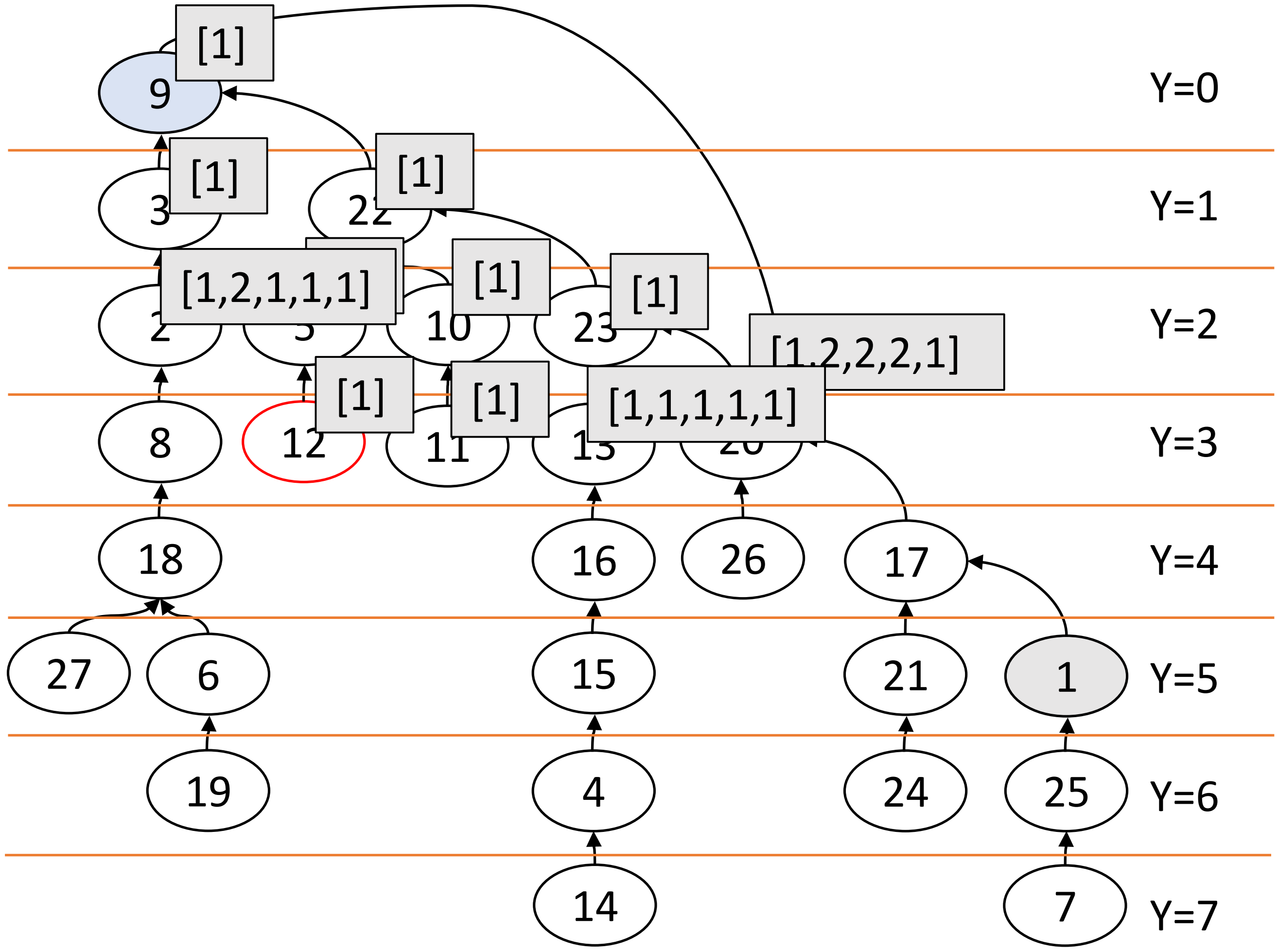
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



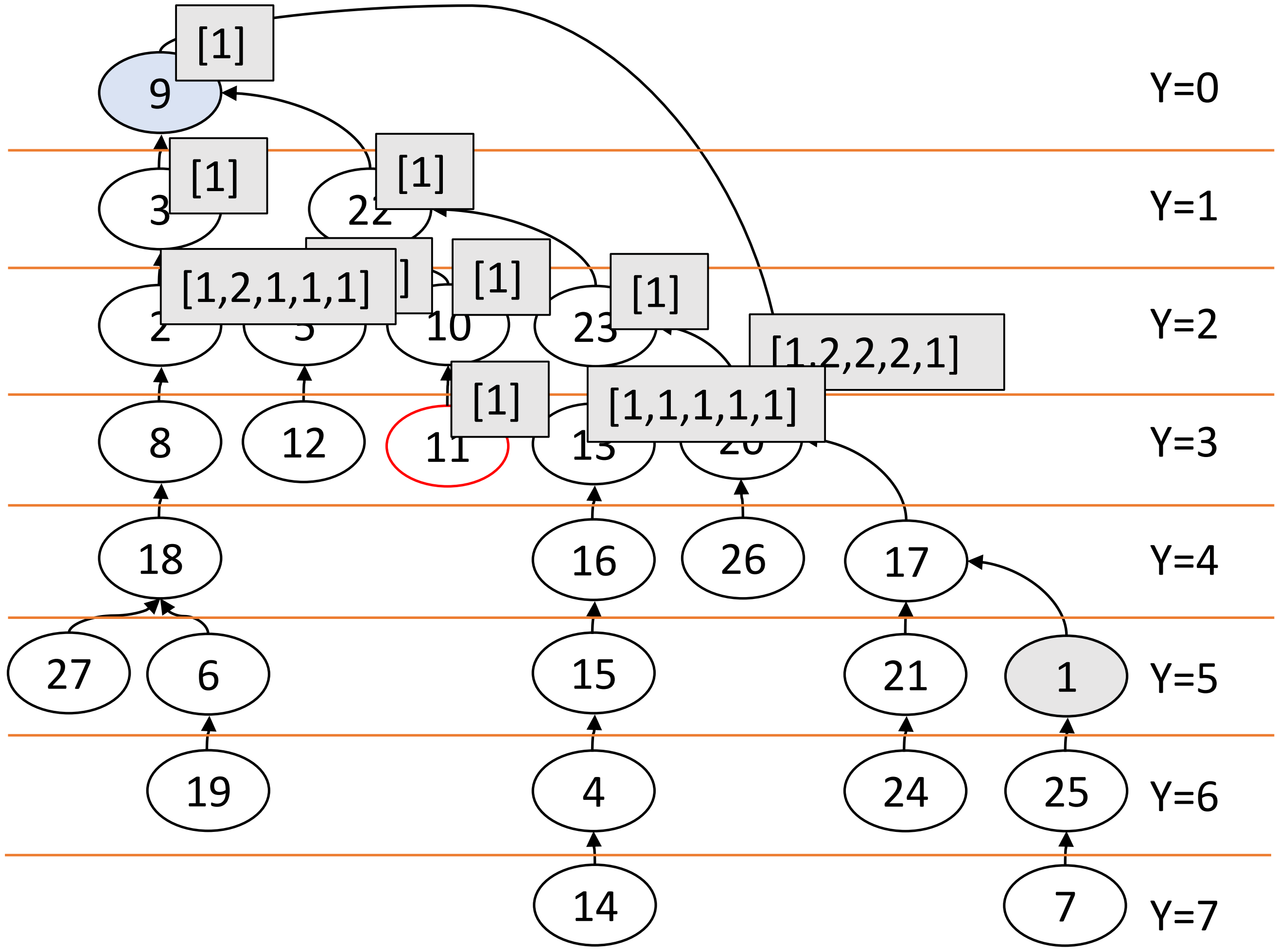
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



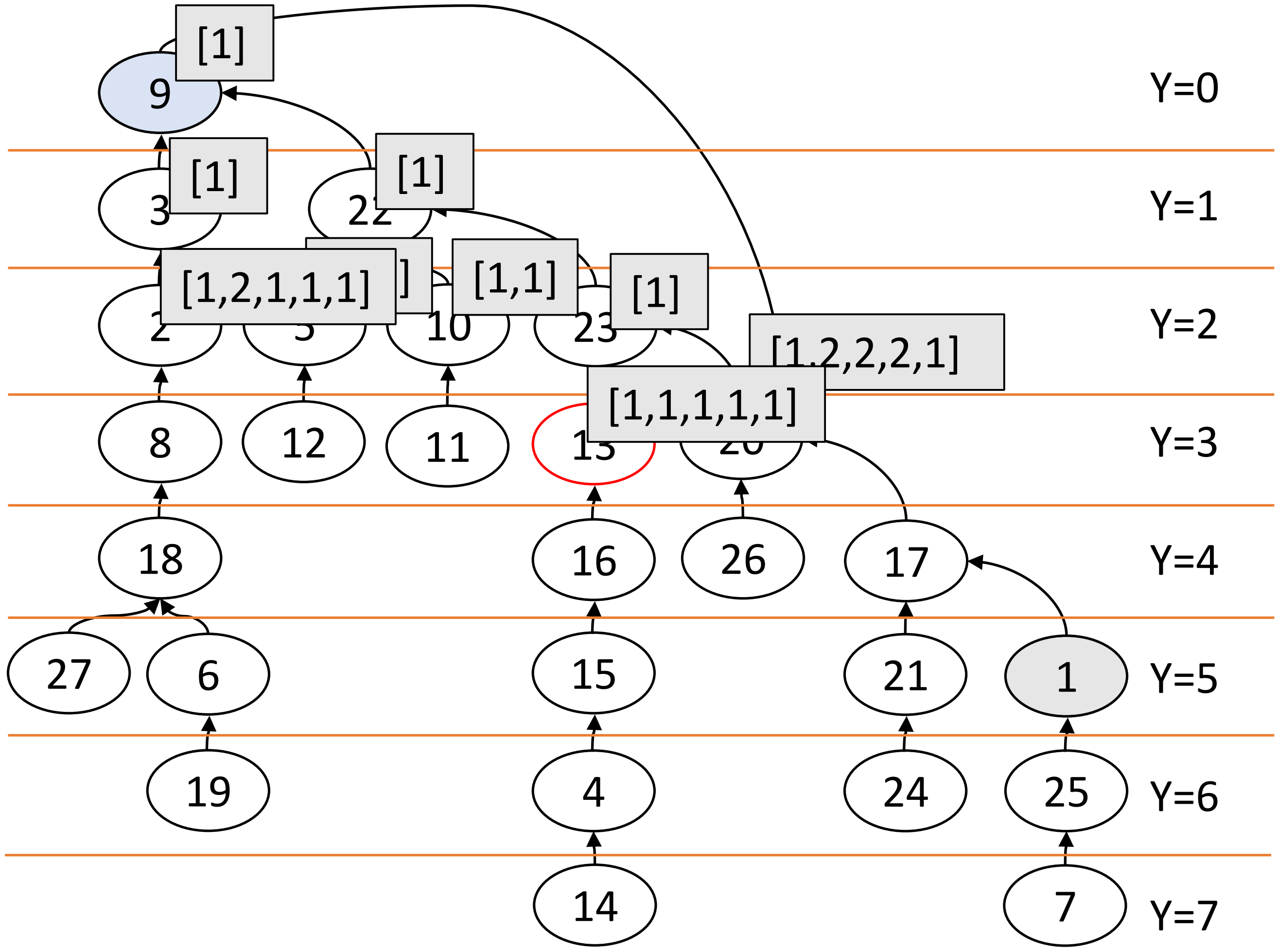
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



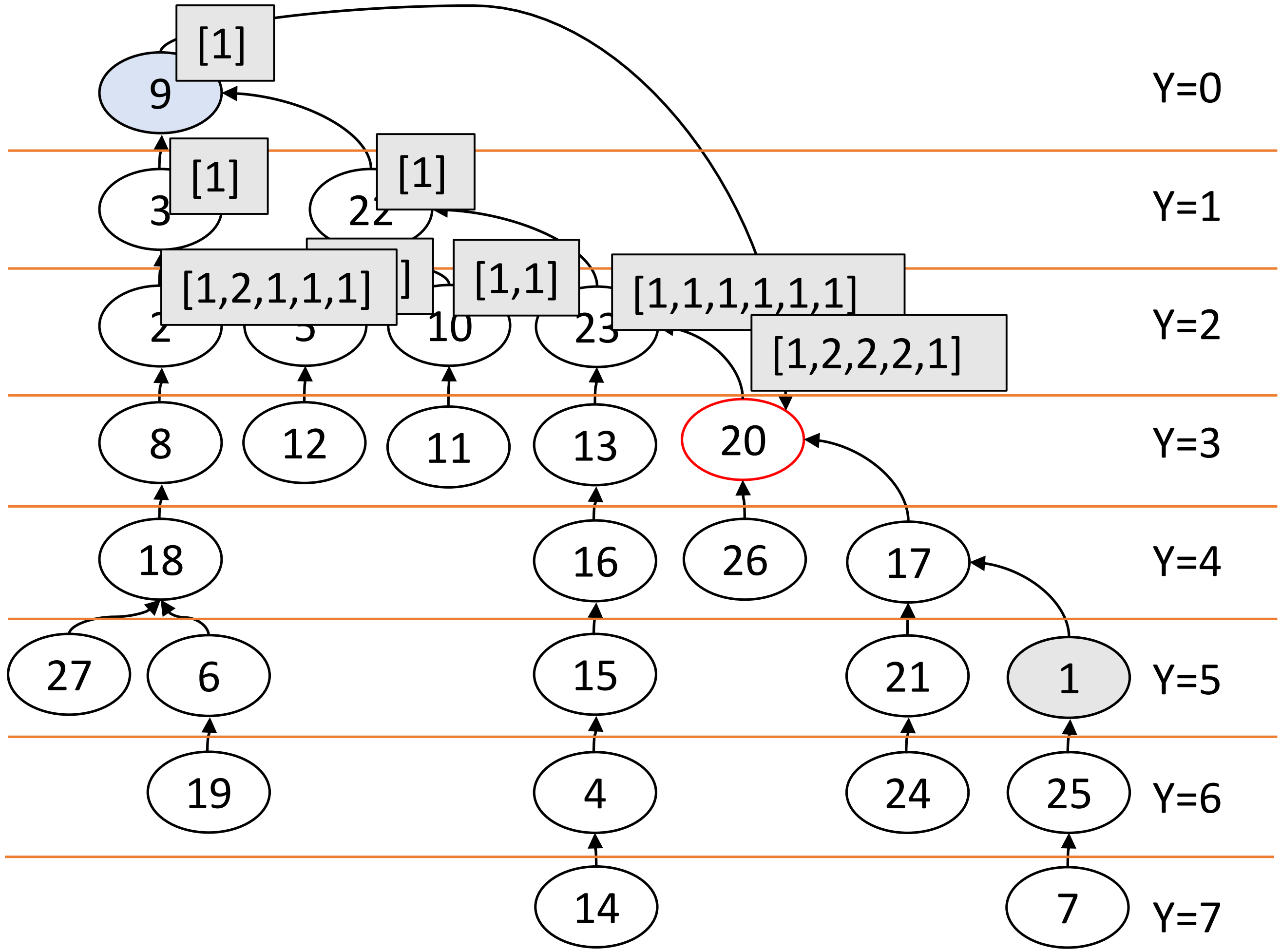
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



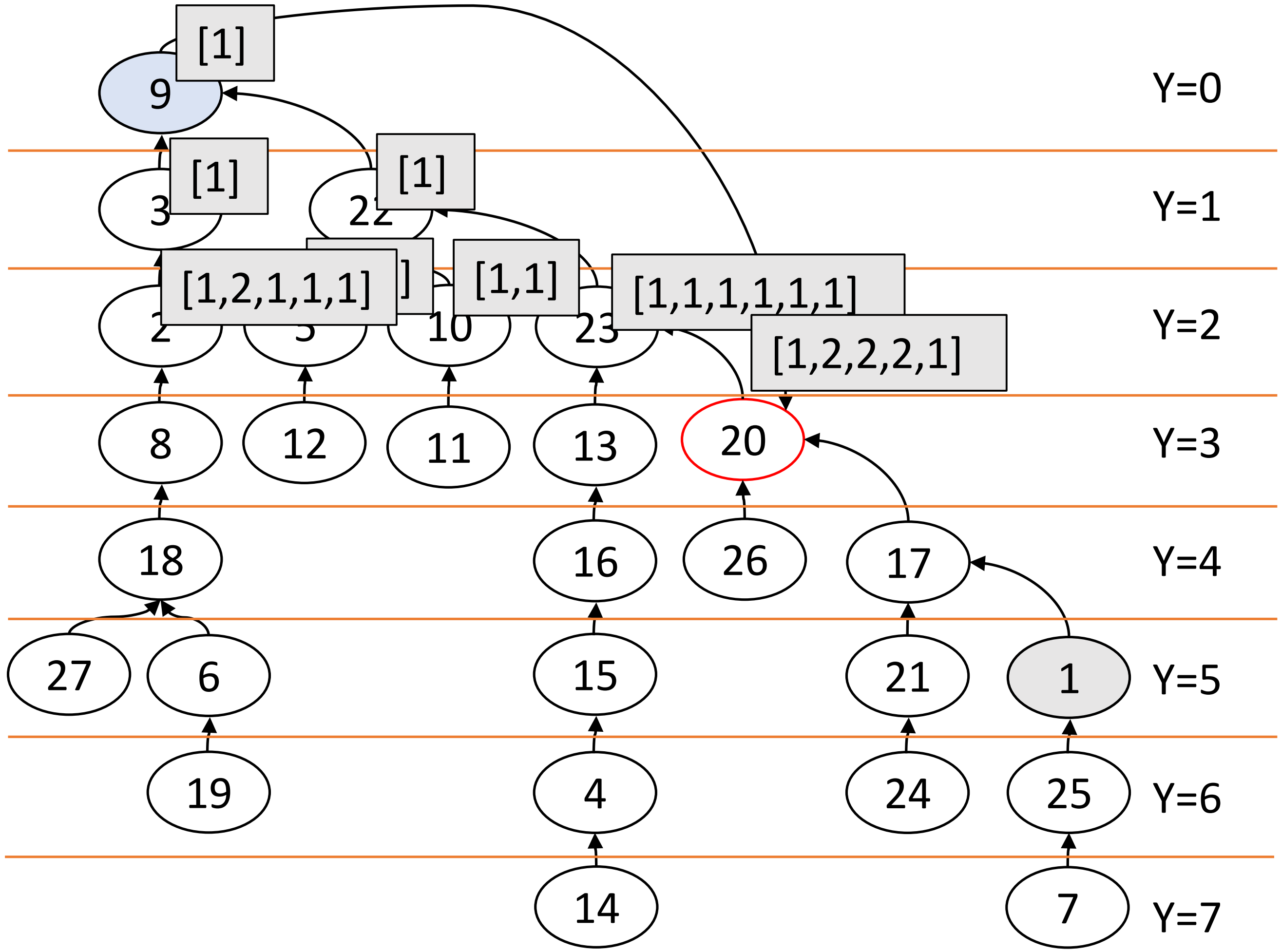
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



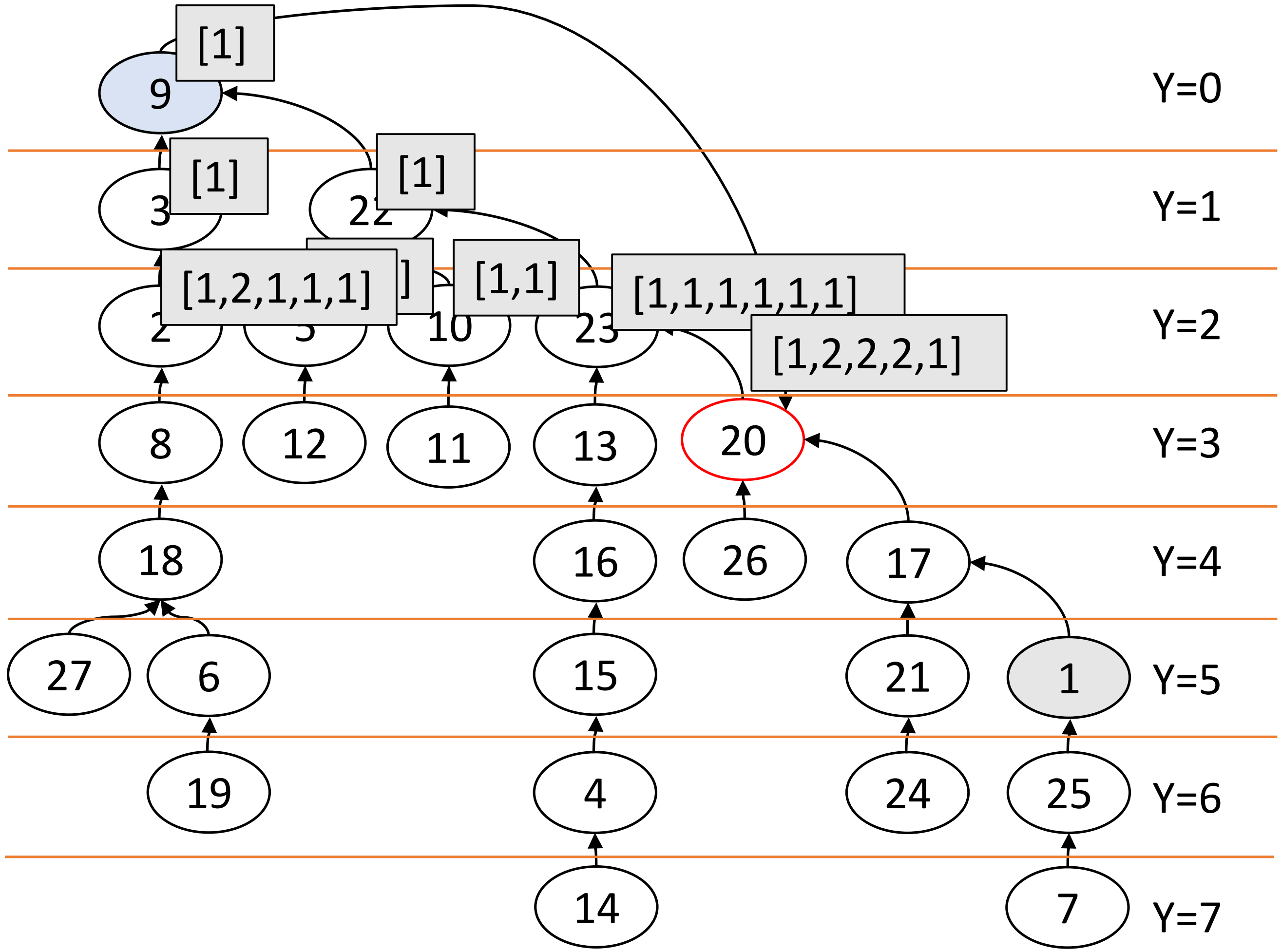
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査
  - この頂点の処理には注意



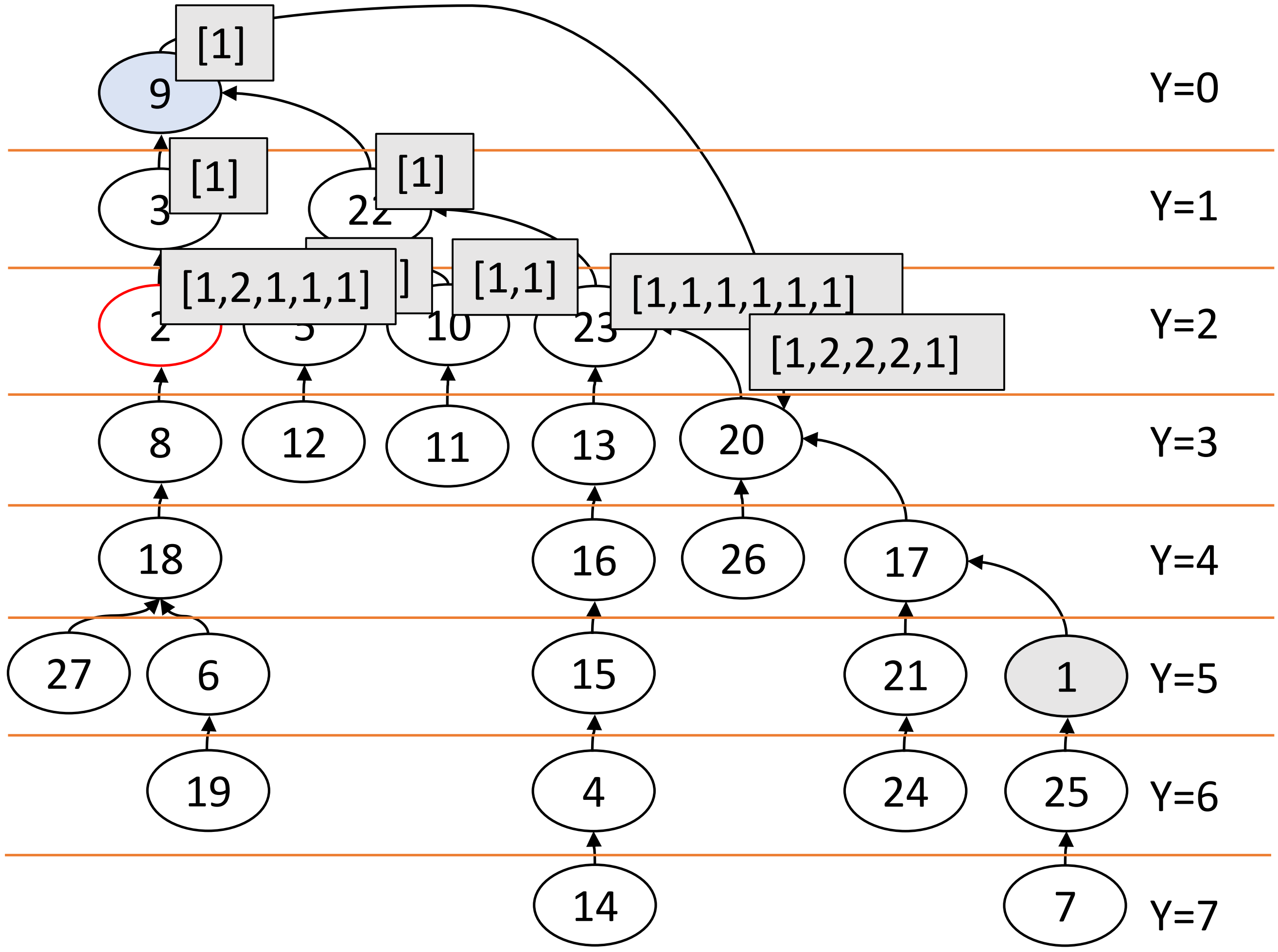
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査
  - この頂点の処理には注意(3B)の場合、この個数は親に遺伝させないようにするとよい



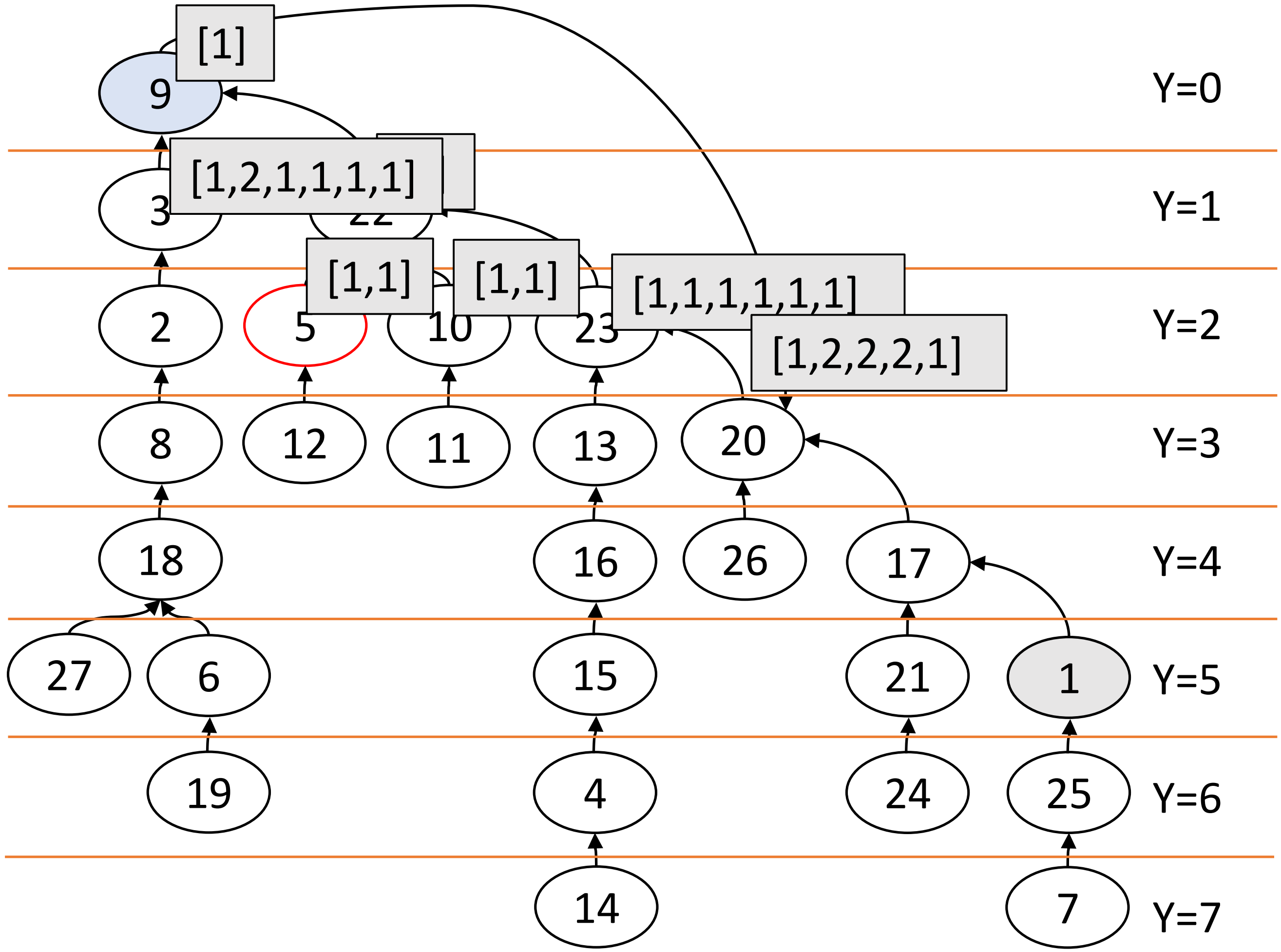
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



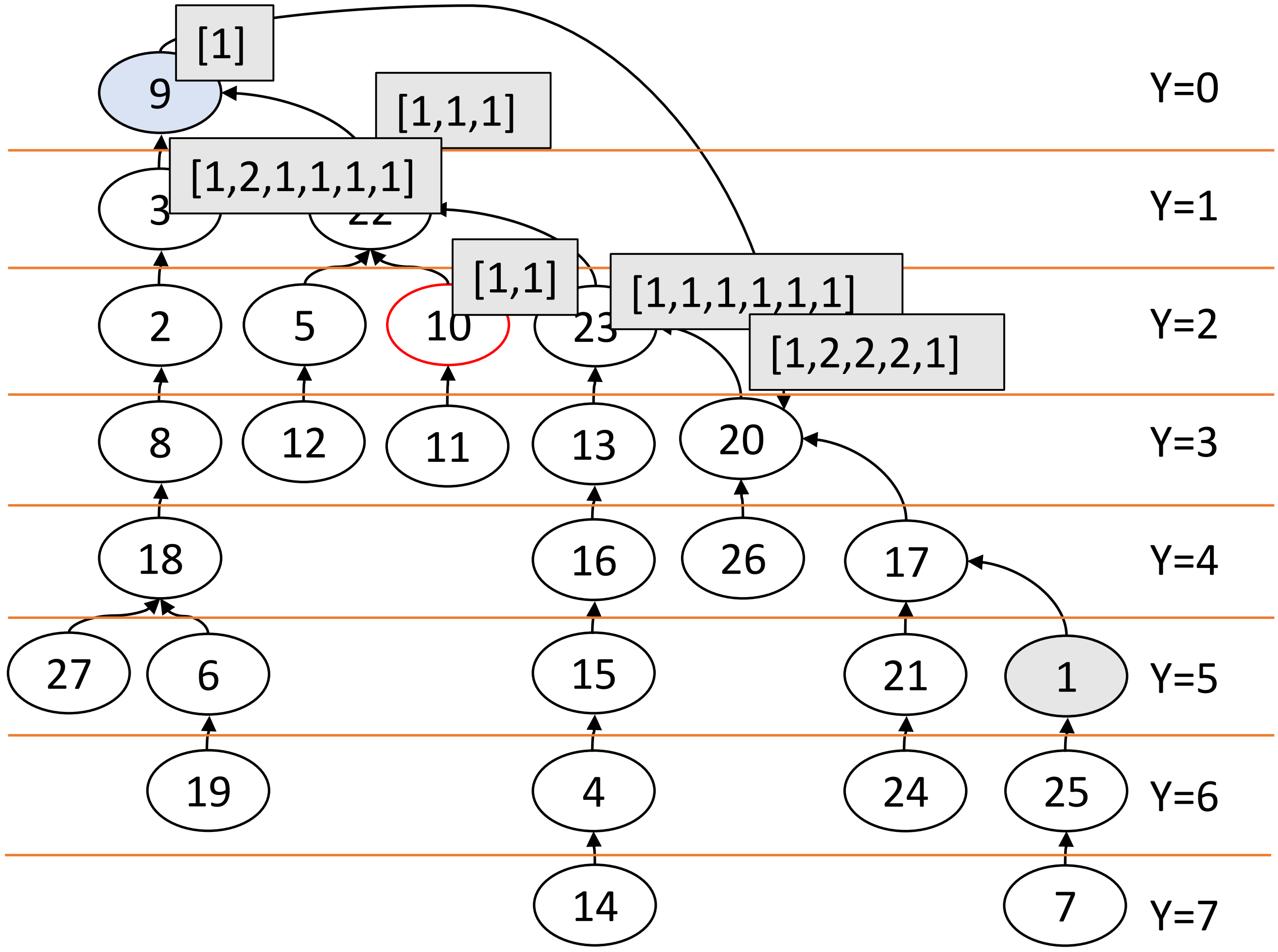
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



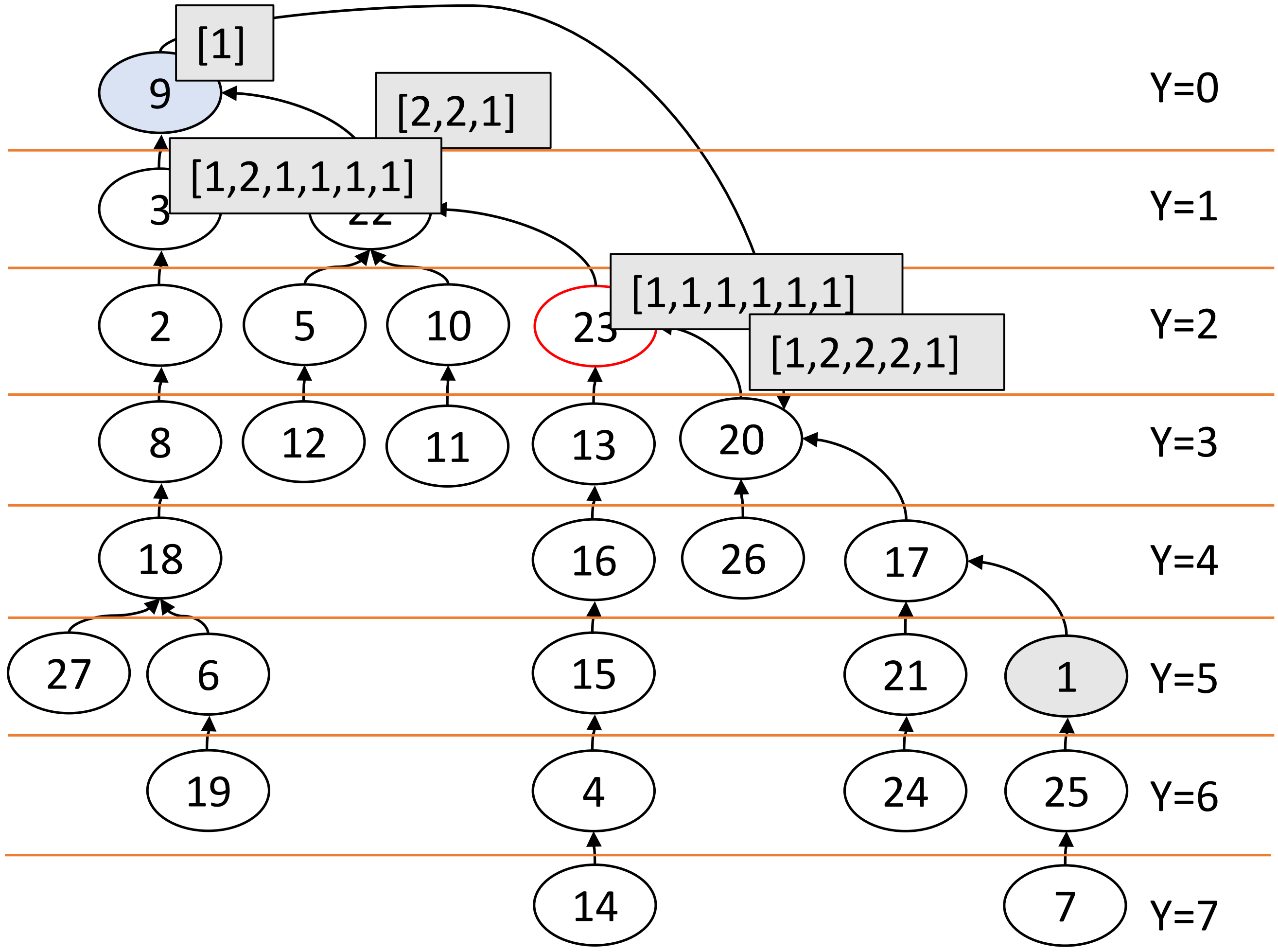
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



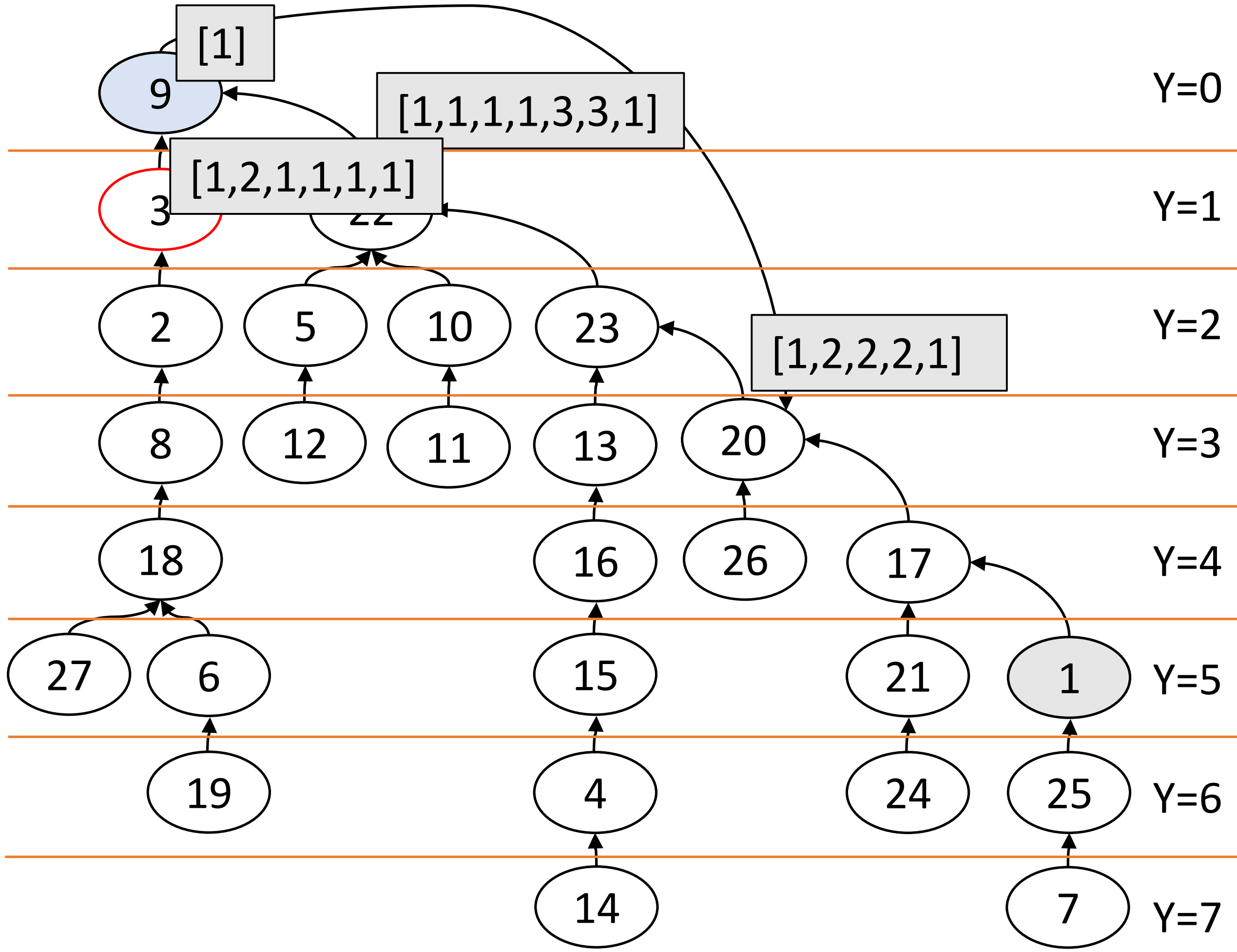
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



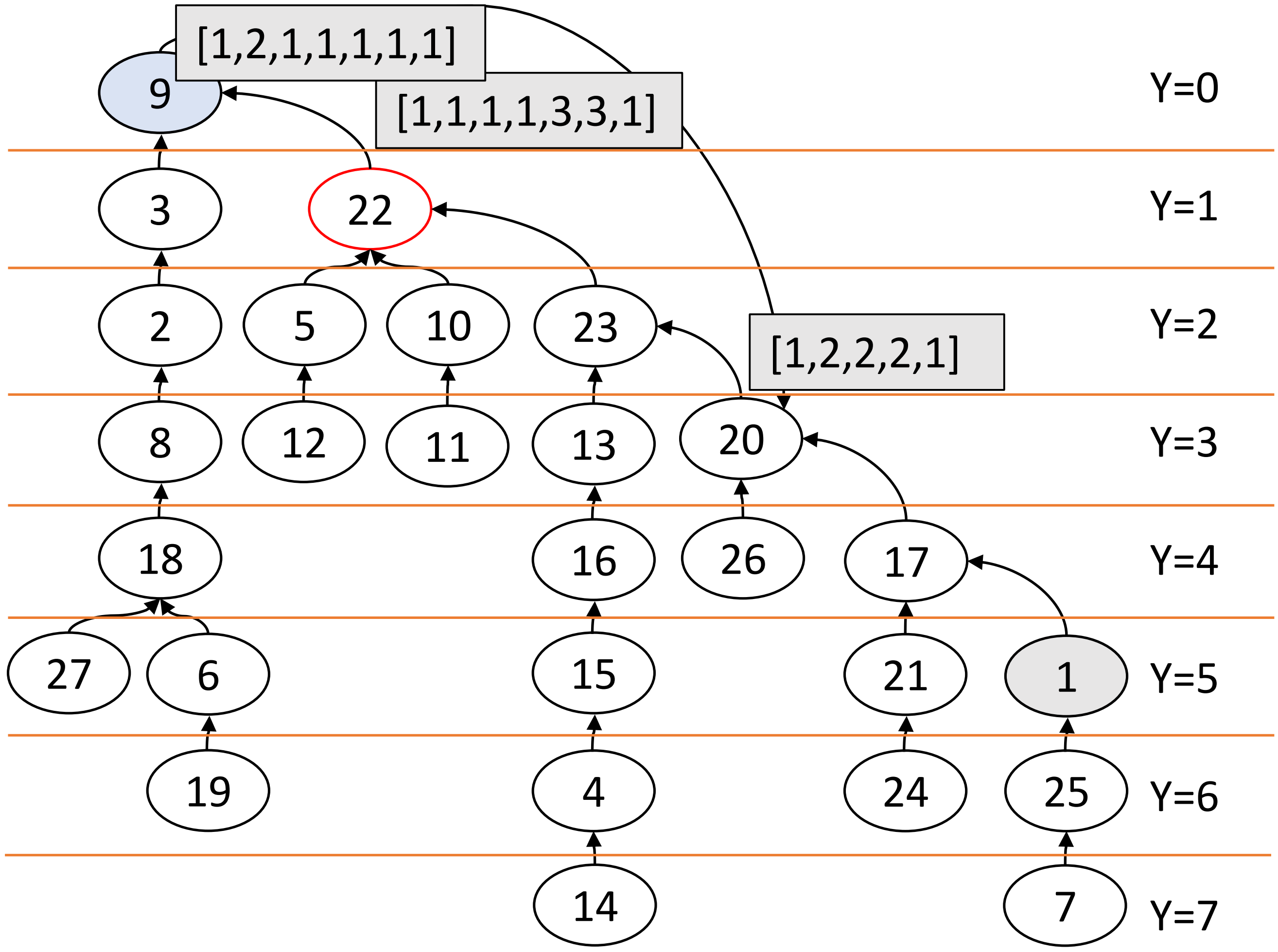
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



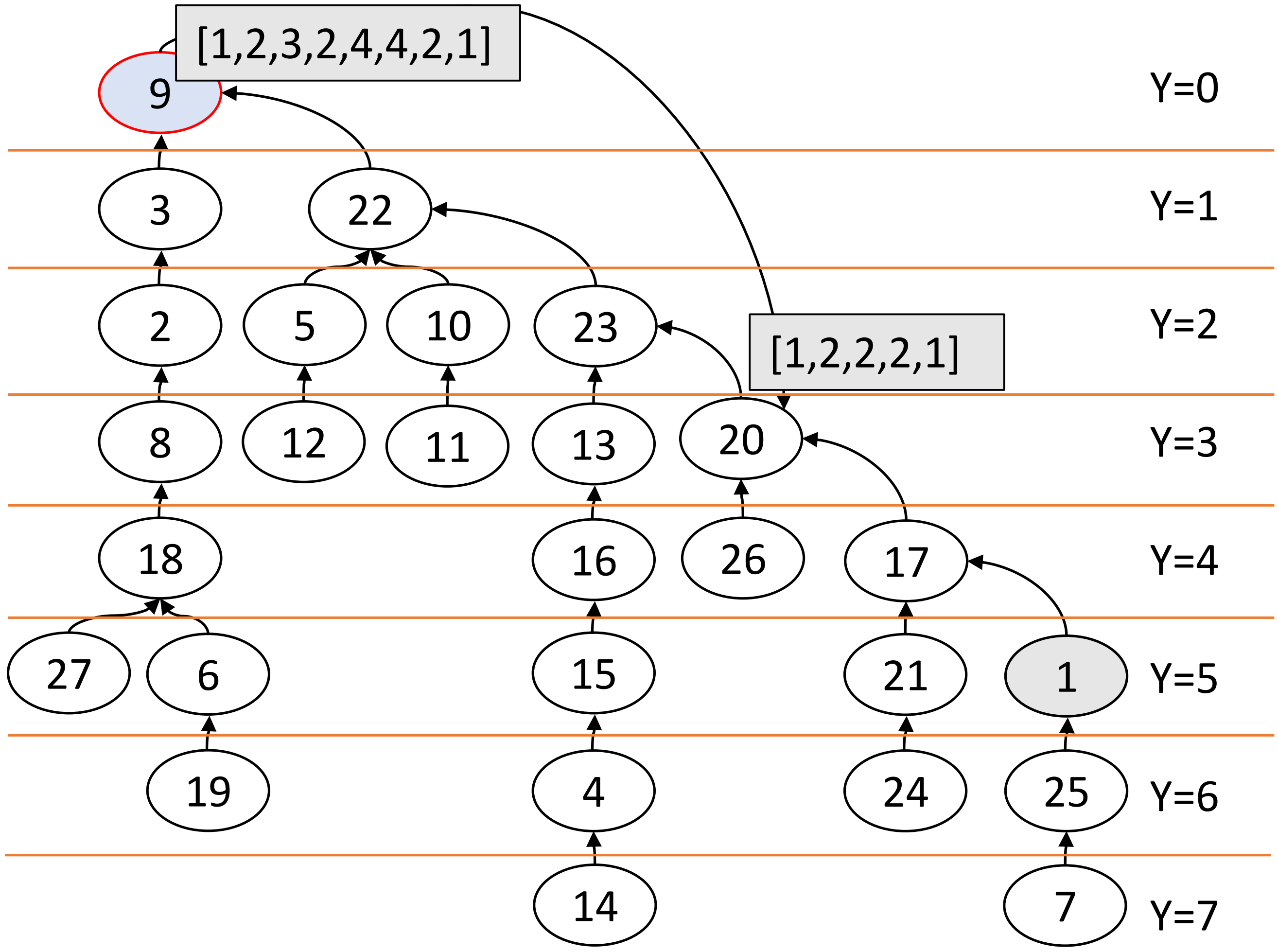
満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



満点解法 – 場合分け(3B)

- Bの候補を高速に数えたい
- 各頂点でFenwick Treeを保持
  - はじめは[1]を表す木として初期化
- イベントを処理しながら下から走査



満点解法 – 場合分け(3B)

- Fenwick Treeのマージ

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする
- 子の末尾に0を付加する [2,5,6,0]

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする
- 子の末尾に0を付加する [2,5,6,0]
- 親の[1,2,3]と[2,5,6,0]を**右揃え**で加算し、新しい列を作る

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする
- 子の末尾に0を付加する [2,5,6,0]
- 親の[1,2,3]と[2,5,6,0]を**右揃え**で加算し、新しい列を作る
- → [2,6,8,3]

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする
- 子の末尾に0を付加する [2,5,6,0]
- 親の[1,2,3]と[2,5,6,0]を**右揃え**で加算し、新しい列を作る
- → [2,6,8,3]
- このとき、**小さい方のデータを大きい方のデータに加える**ようにする

満点解法 – 場合分け(3B)

- Fenwick Treeのマージ
- 親が [1,2,3] 、子が [2,5,6] というデータを保持していたとする
- 子の末尾に0を付加する [2,5,6,0]
- 親の[1,2,3]と[2,5,6,0]を**右揃え**で加算し、新しい列を作る
- → [2,6,8,3]
- このとき、**小さい方のデータを大きい方のデータに加える**ようにする
- → データ構造をマージする一般的なテク。全体で $O(n \log\ n)$
  - 対数は2乗にはならない

満点解法 – 場合分け(3B)

- イベント処理の計算量

満点解法 – 場合分け(3B)

- イベント処理の計算量
- イベント : 到達先候補ごとに処理していた

満点解法 – 場合分け(3B)

- イベント処理の計算量
- イベント : 到達先候補ごとに処理していた
- 到達先候補ごとのイベント数が均等ならば、イベント処理回数は $O(n)$

満点解法 – 場合分け(3B)

- イベント処理の計算量
- イベント : 到達先候補ごとに処理していた
- 到達先候補ごとのイベント数が均等ならば、イベント処理回数は $O(n)$
- 1つのイベントは$O(\log n)$で処理できるから、$O(n \log n)$

満点解法 – 場合分け(3B)

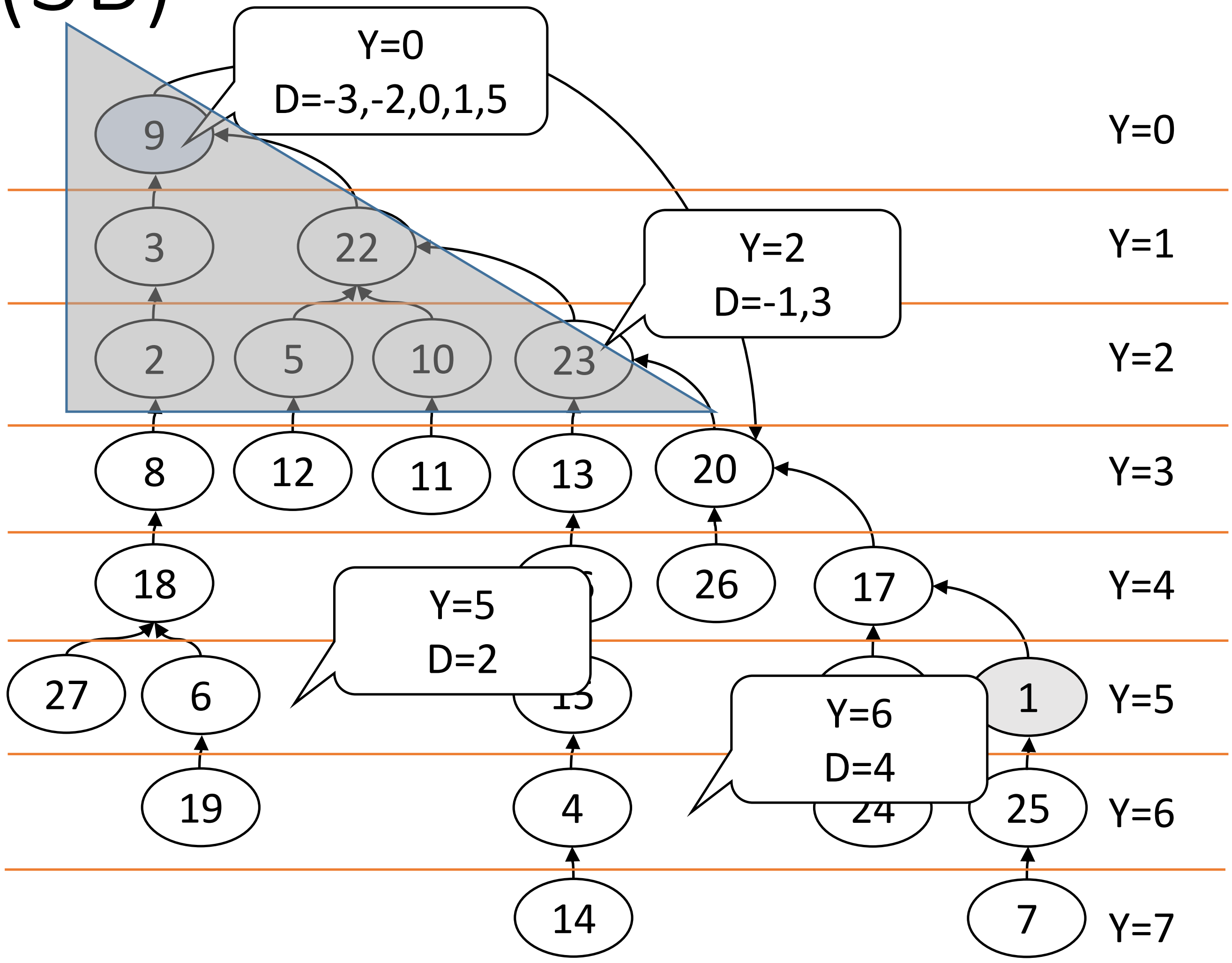
- イベント処理の計算量
- イベント：到達先候補ごとに処理していた
- 到達先候補ごとのイベント数が**均等ならば**、イベント処理回数は $O(n)$
- 1つのイベントは$O(\log n)$で処理できるから、$O(n\log n)$

満点解法 – 場合分け(3B)

- こういうケースがありそう:
  - Y=2にたくさんの頂点がある　かつ
  - Y=2にたくさんのイベントがある
- この場合、イベント処理回数は$O(n^2)$になりかねない

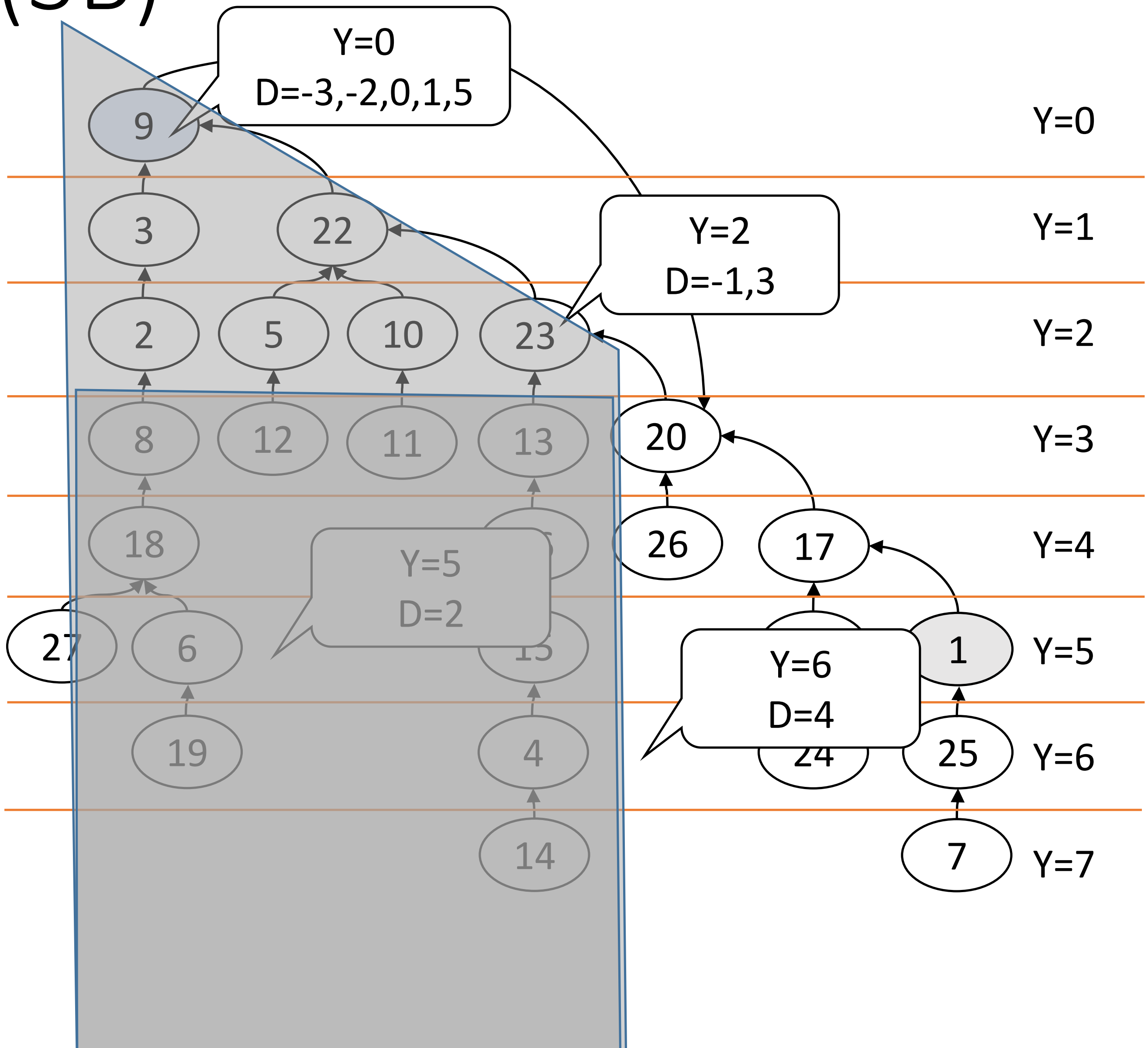
満点解法 – 場合分け(3B)

- イベントの図を思い出そう



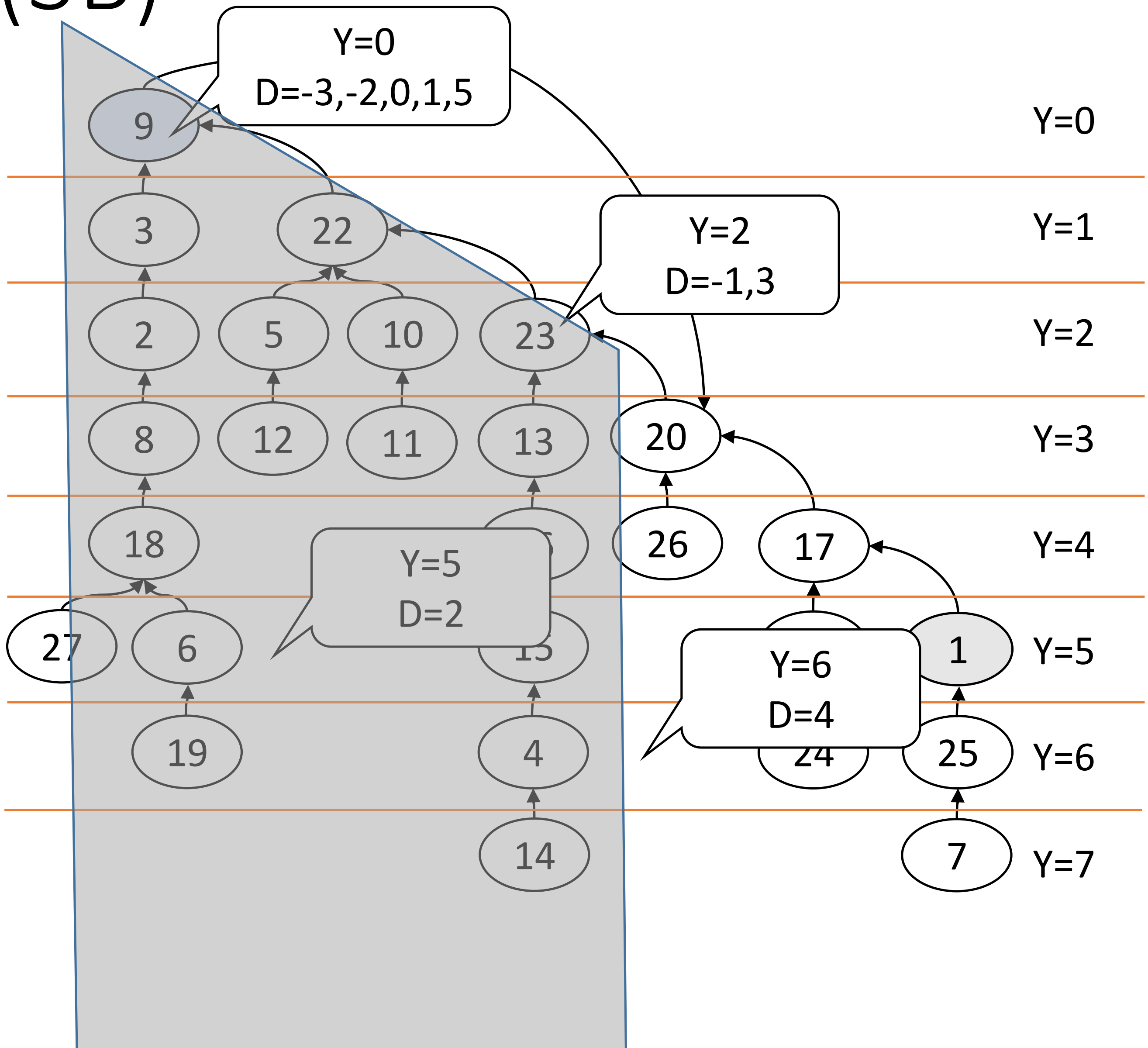
満点解法 – 場合分け(3B)

- 簡単のため、数え上げ処理を2つに分割



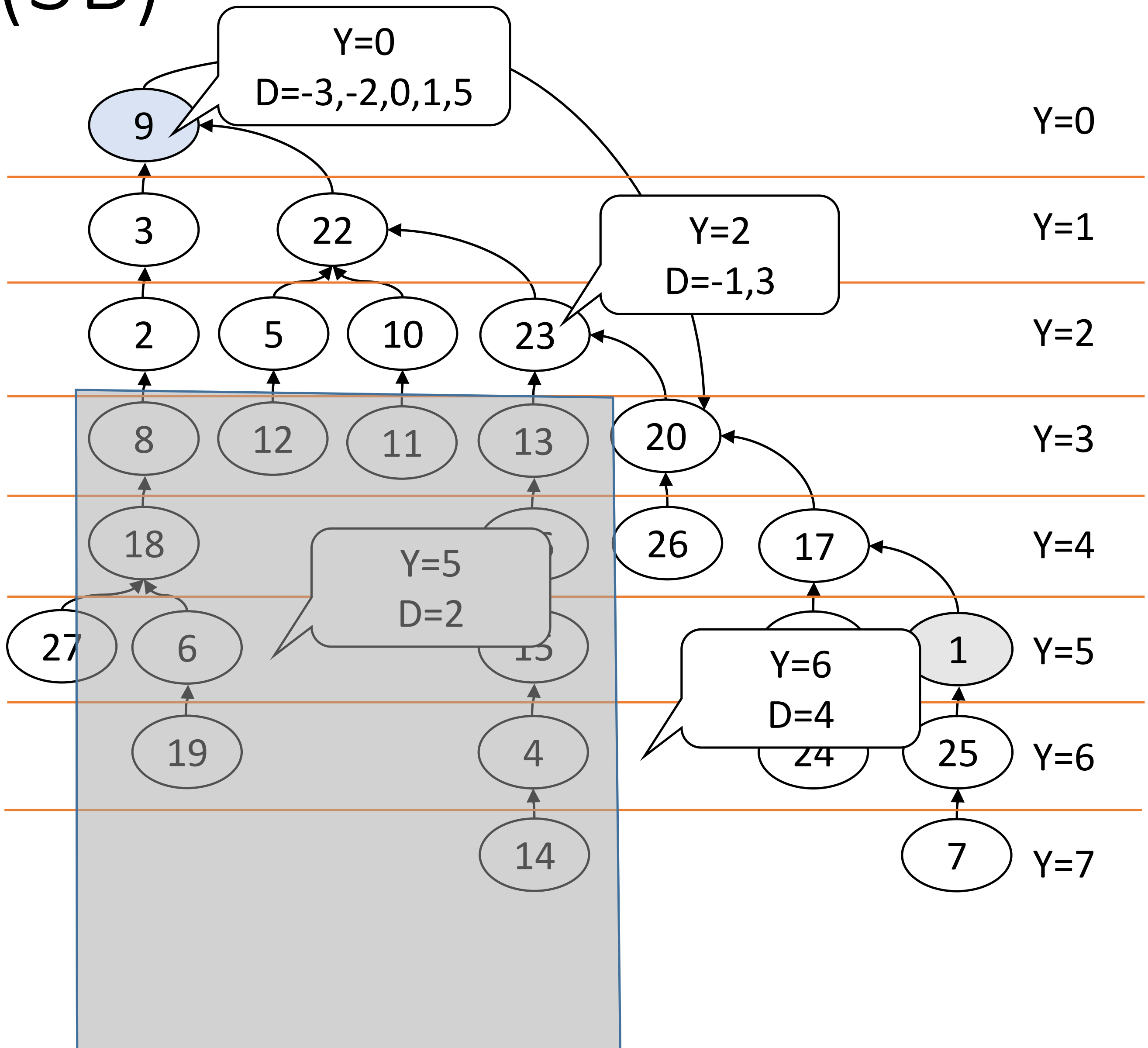
満点解法 – 場合分け(3B)

- 簡単のため、数え上げ処理を2つに分割
  - 加算処理



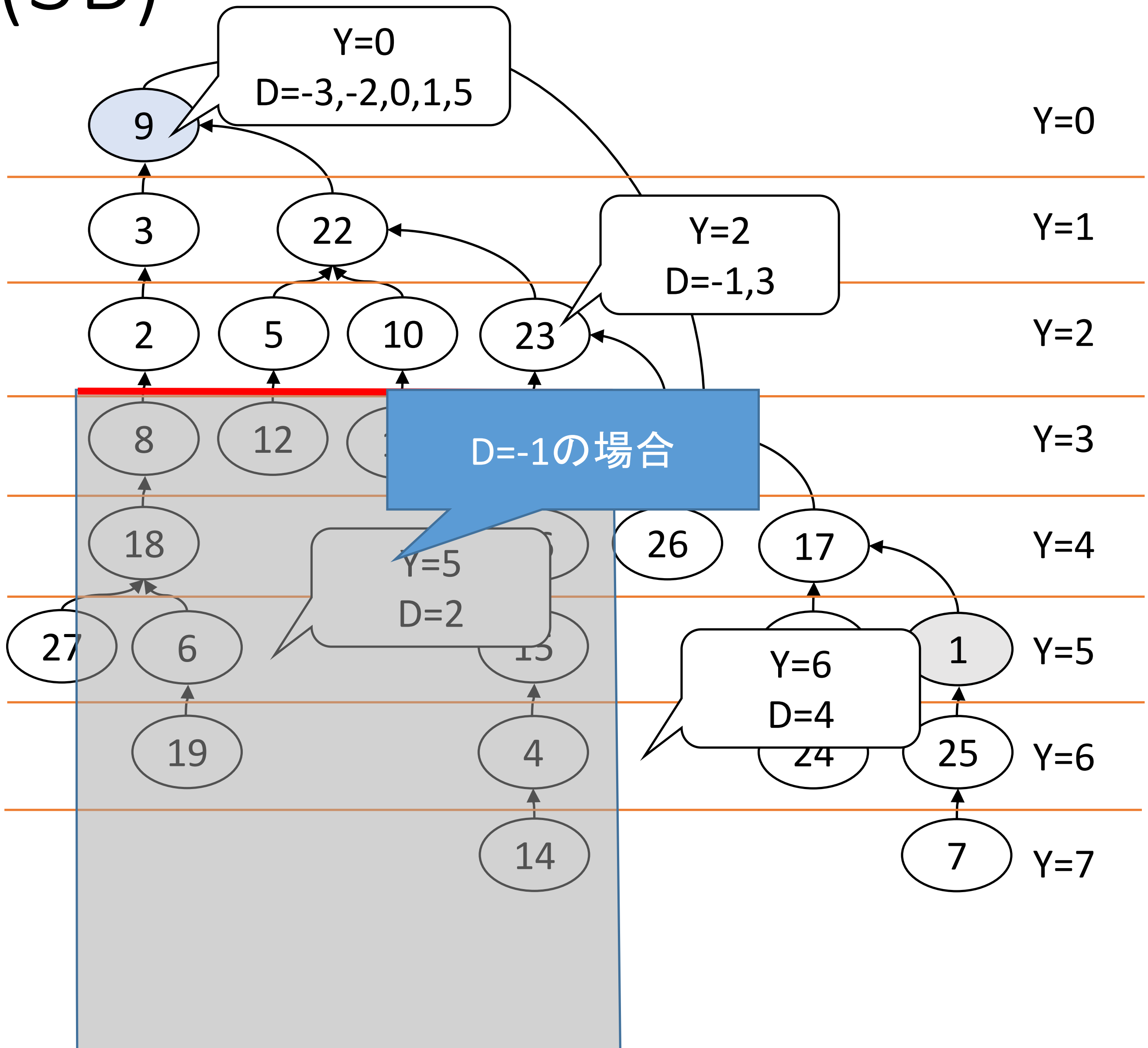
満点解法 – 場合分け(3B)

- 簡単のため、数え上げ処理を2つに分割
  - 加算処理
  - 減算処理



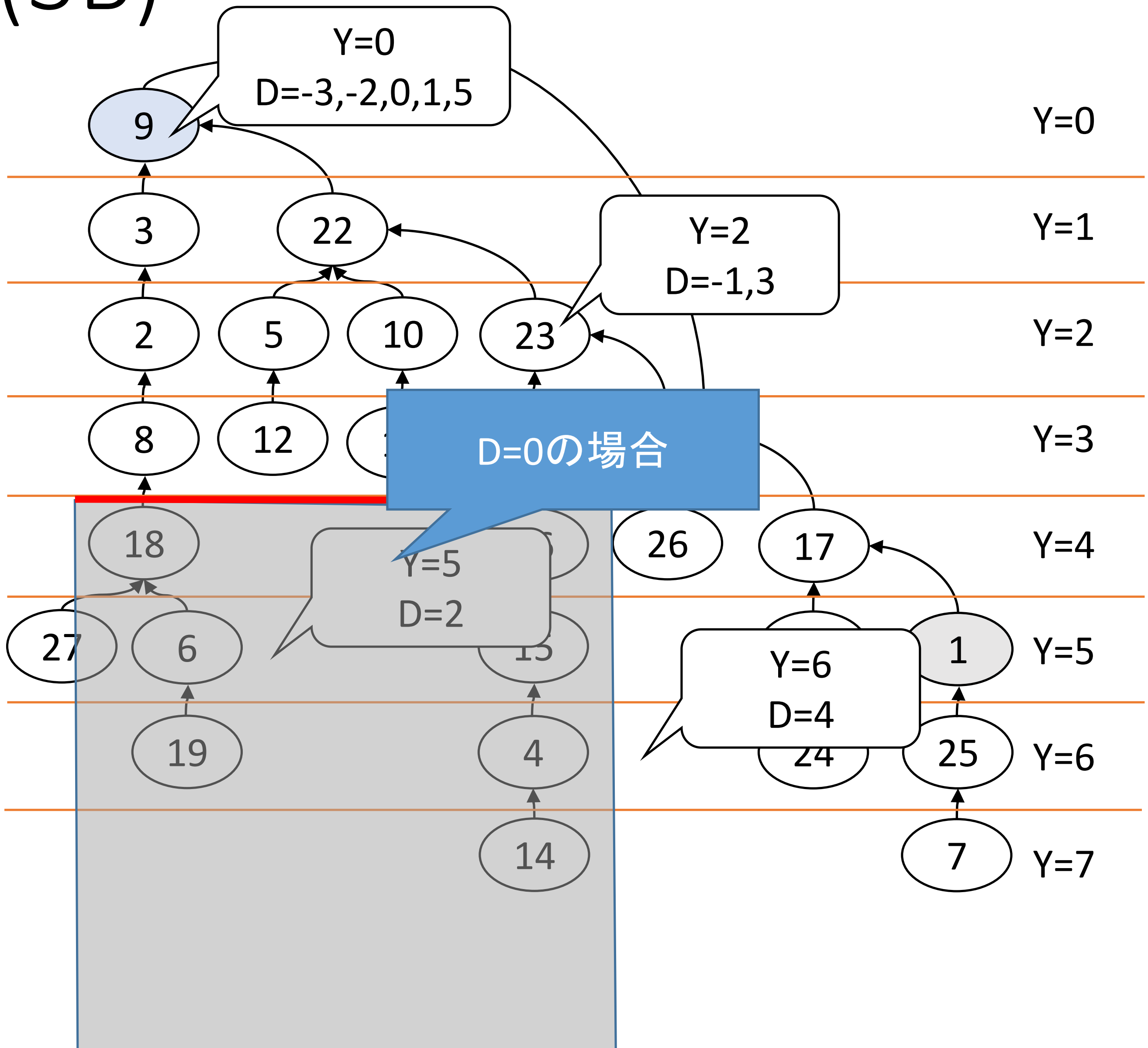
満点解法 – 場合分け(3B)

- 各イベントの数え上げ範囲はDが変化すると上下に動く



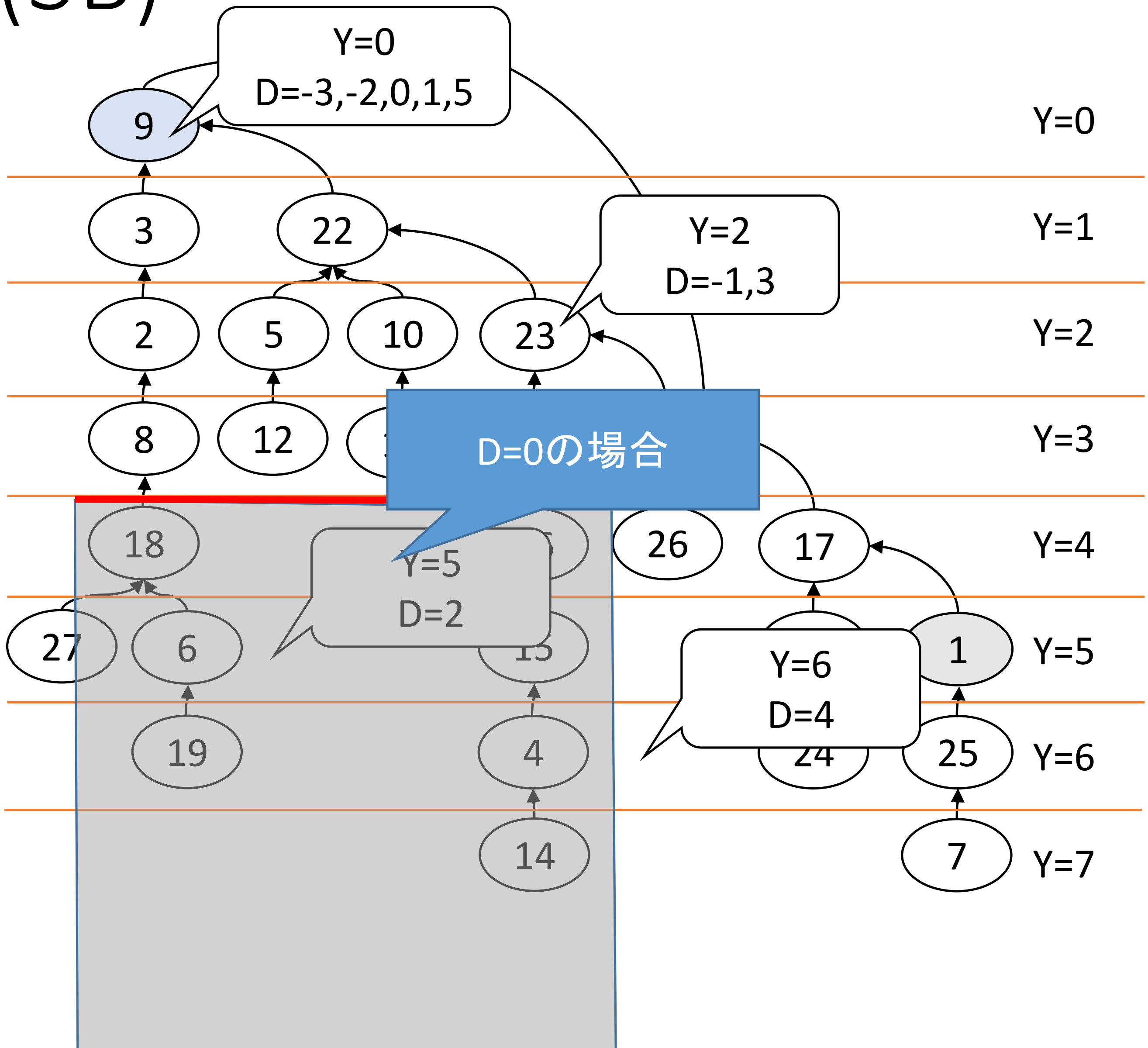
満点解法 – 場合分け(3B)

- 各イベントの数え上げ範囲はDが変化すると上下に動く



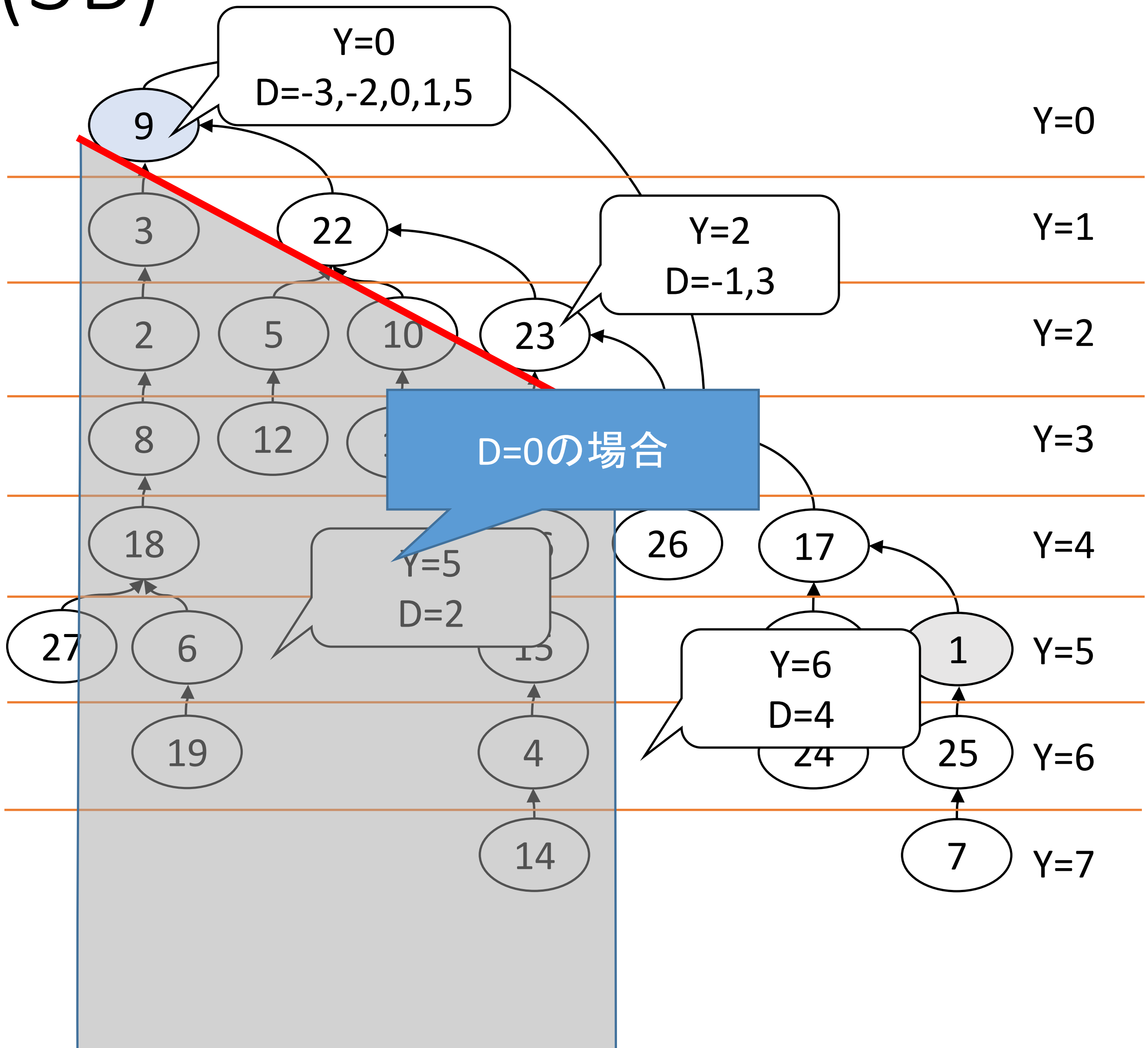
満点解法 – 場合分け(3B)

- 各イベントの境界部(赤線で示した)に注目する



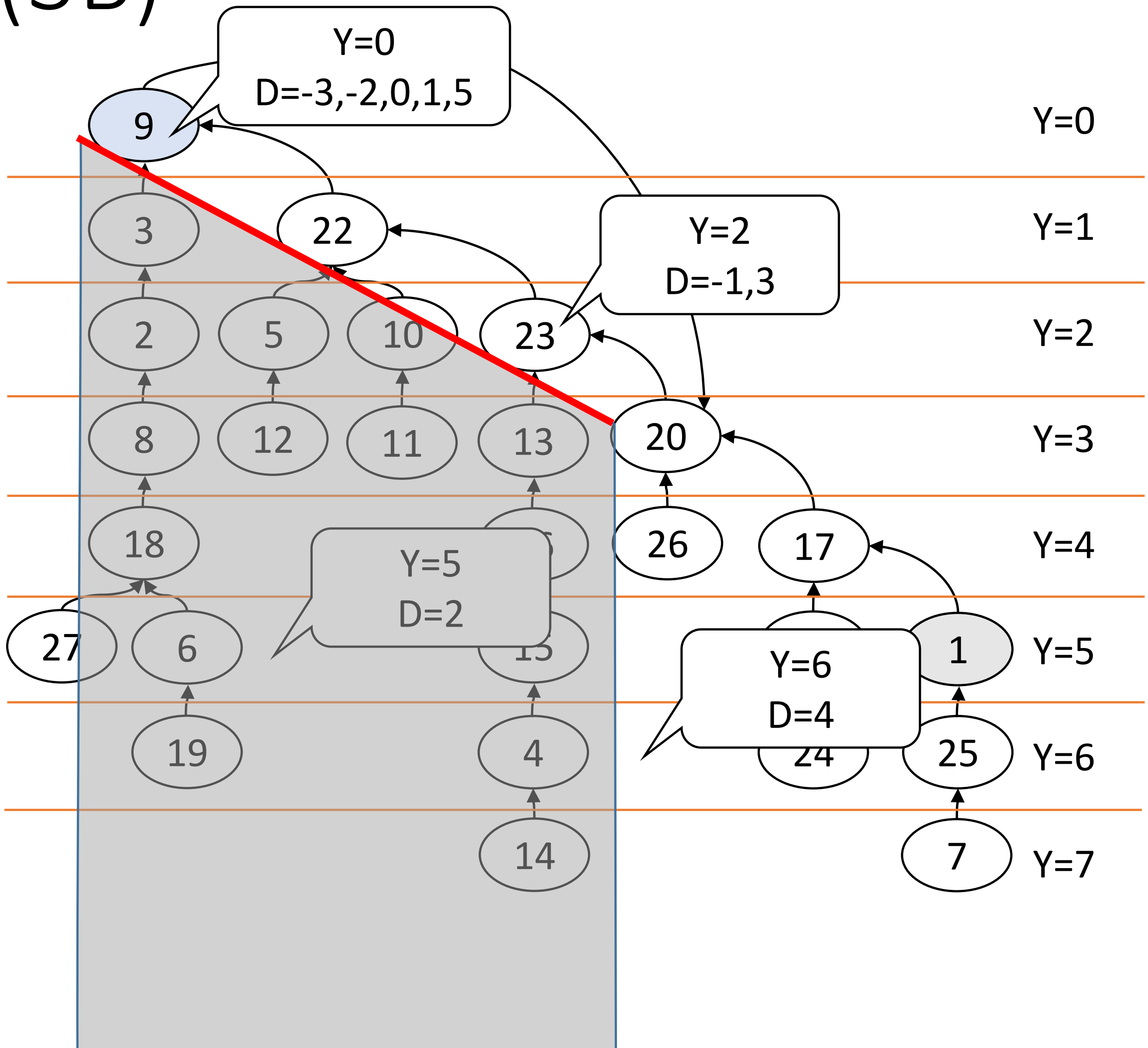
満点解法 – 場合分け(3B)

- 各イベントの境界部
  (赤線で示した)に注目する
  - 加算処理の場合もこんな感じで
    同様



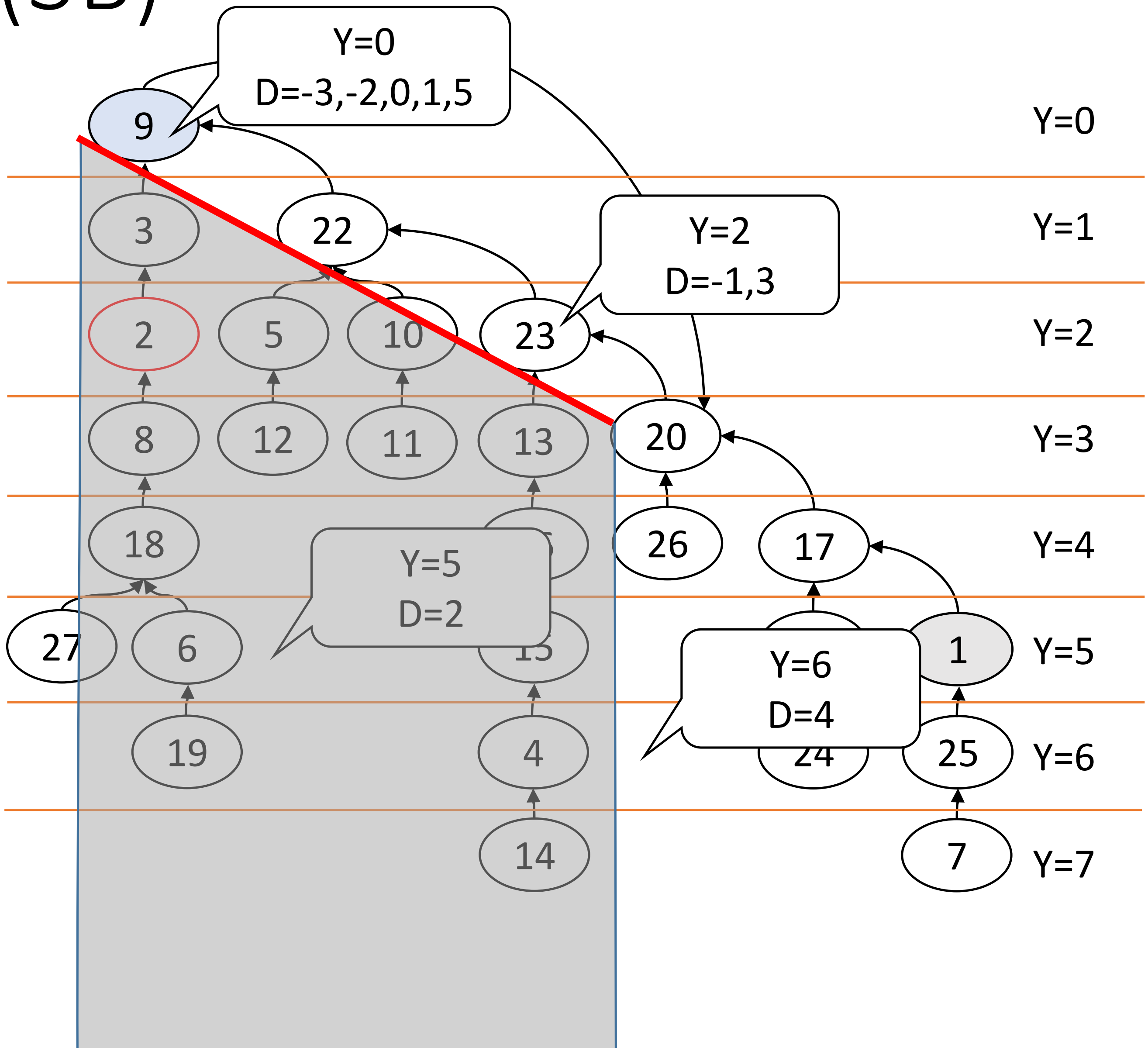
満点解法 – 場合分け(3B)

- 各イベントの境界部(赤線で示した)に注目する
- 意味があるものだけ処理したい



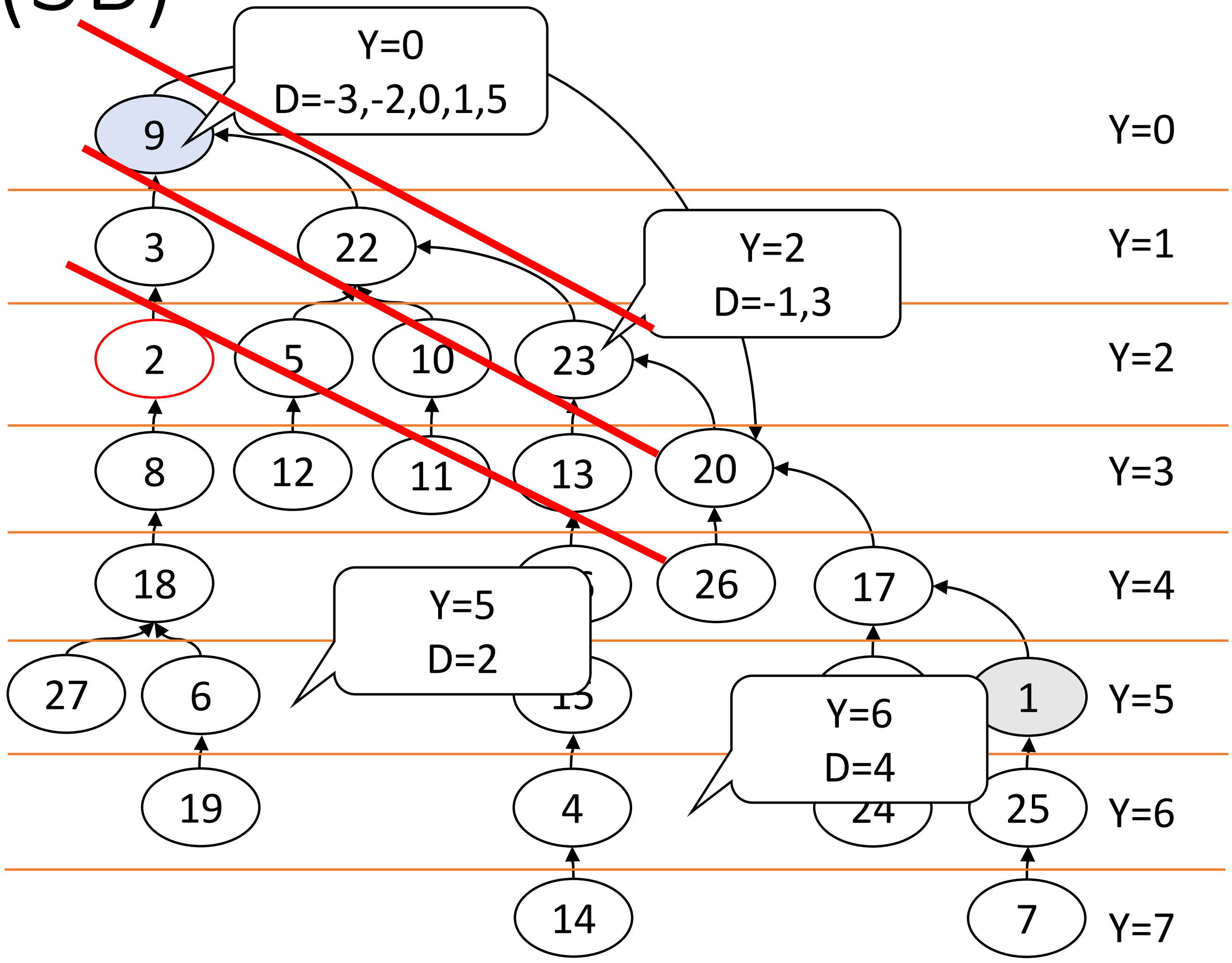
満点解法 – 場合分け(3B)

- 各イベントの境界部
  (赤線で示した)に注目する
- 意味があるものだけ処理したい
- 例えば、2を処理中のとき



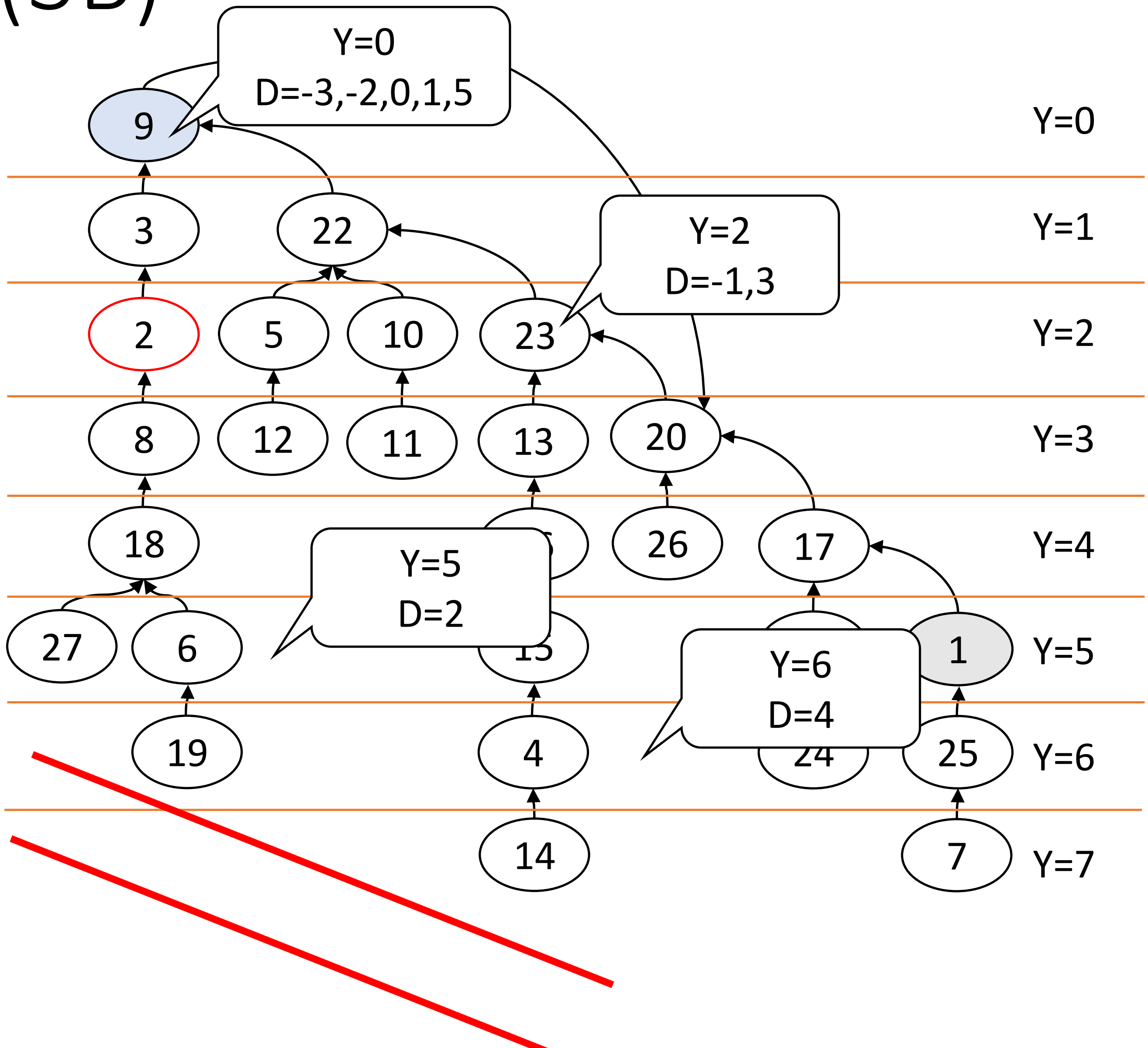
満点解法 – 場合分け(3B)

- 各イベントの境界部(赤線で示した)に注目する
- 意味があるものだけ処理したい
- 例えば、2を処理中のとき
  - こういった境界をもつイベントはたくさんあっても一括処理できる
  - (全て、数え上げの結果が一緒)



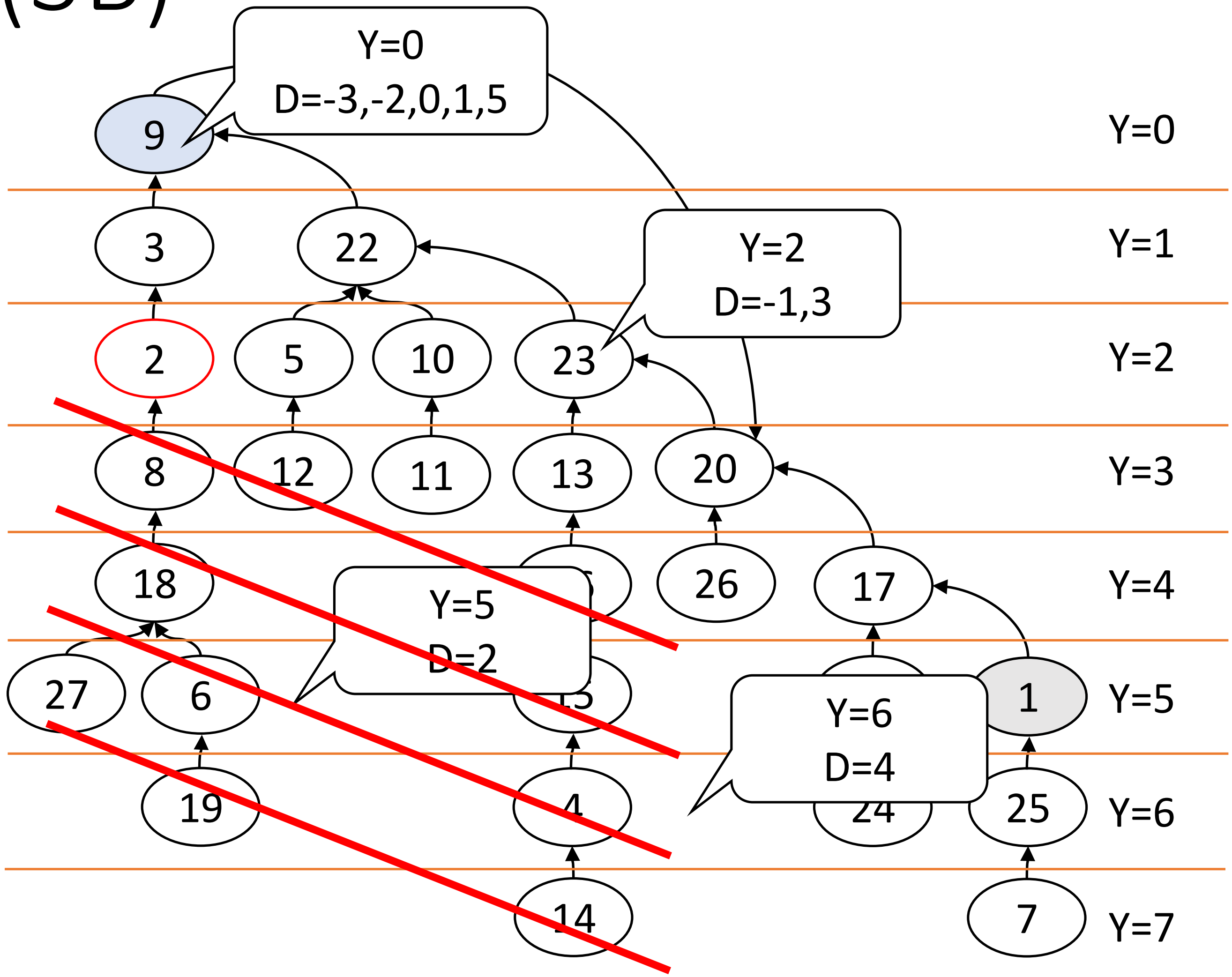
満点解法 – 場合分け(3B)

- 各イベントの境界部
  (赤線で示した)に注目する
- 意味があるものだけ処理したい
- 例えば、2を処理中のとき
  - 逆に、こういった境界をもつイベントはたくさんあってもいずれも無視してよい
  - (全て、数え上げの結果は0)



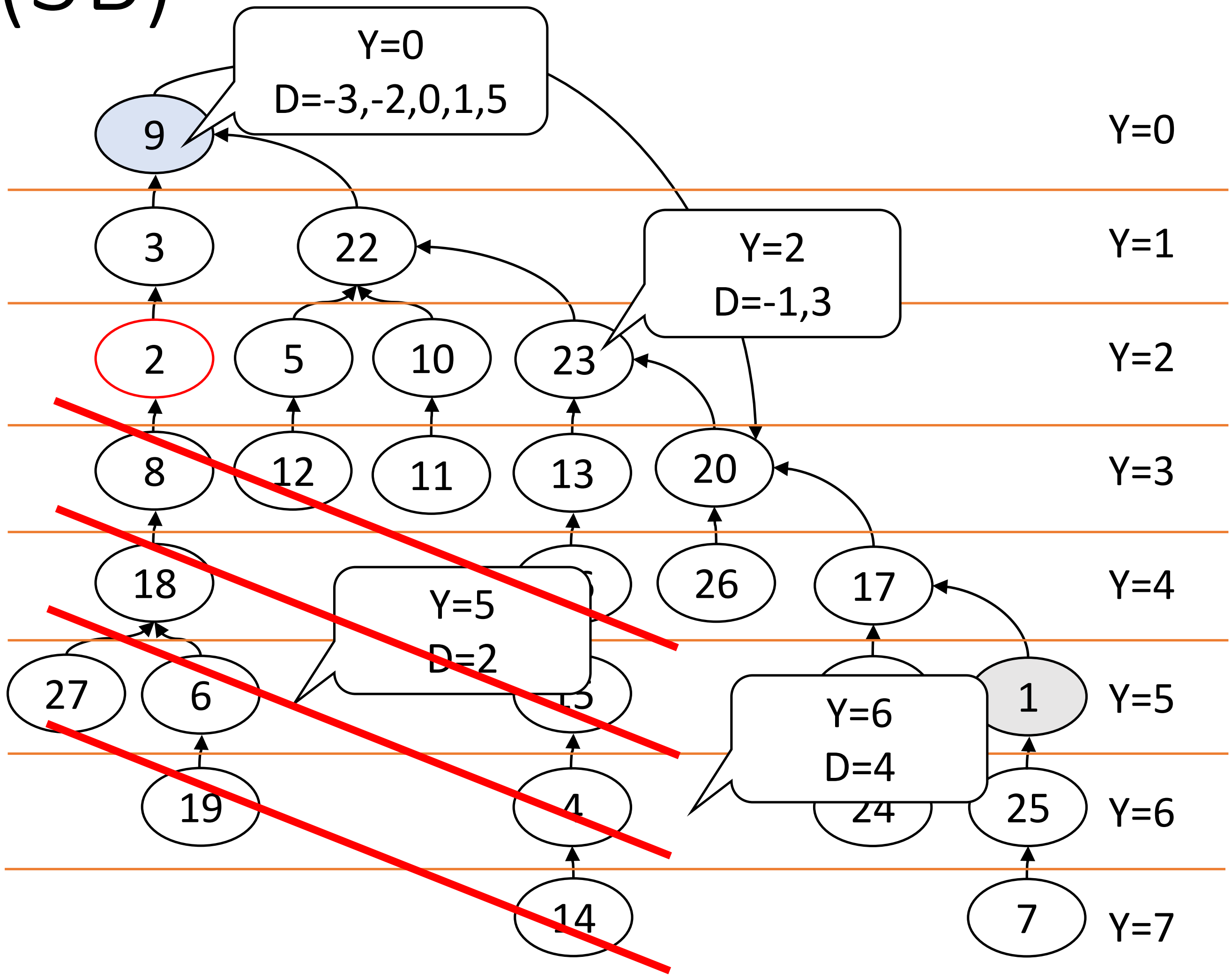
満点解法 – 場合分け(3B)

- 各イベントの境界部
  (赤線で示した)に注目する
- 意味があるものだけ処理したい
- 例えば、2を処理中のとき
  - 境界がこの範囲にあるイベントだけを処理すればよい



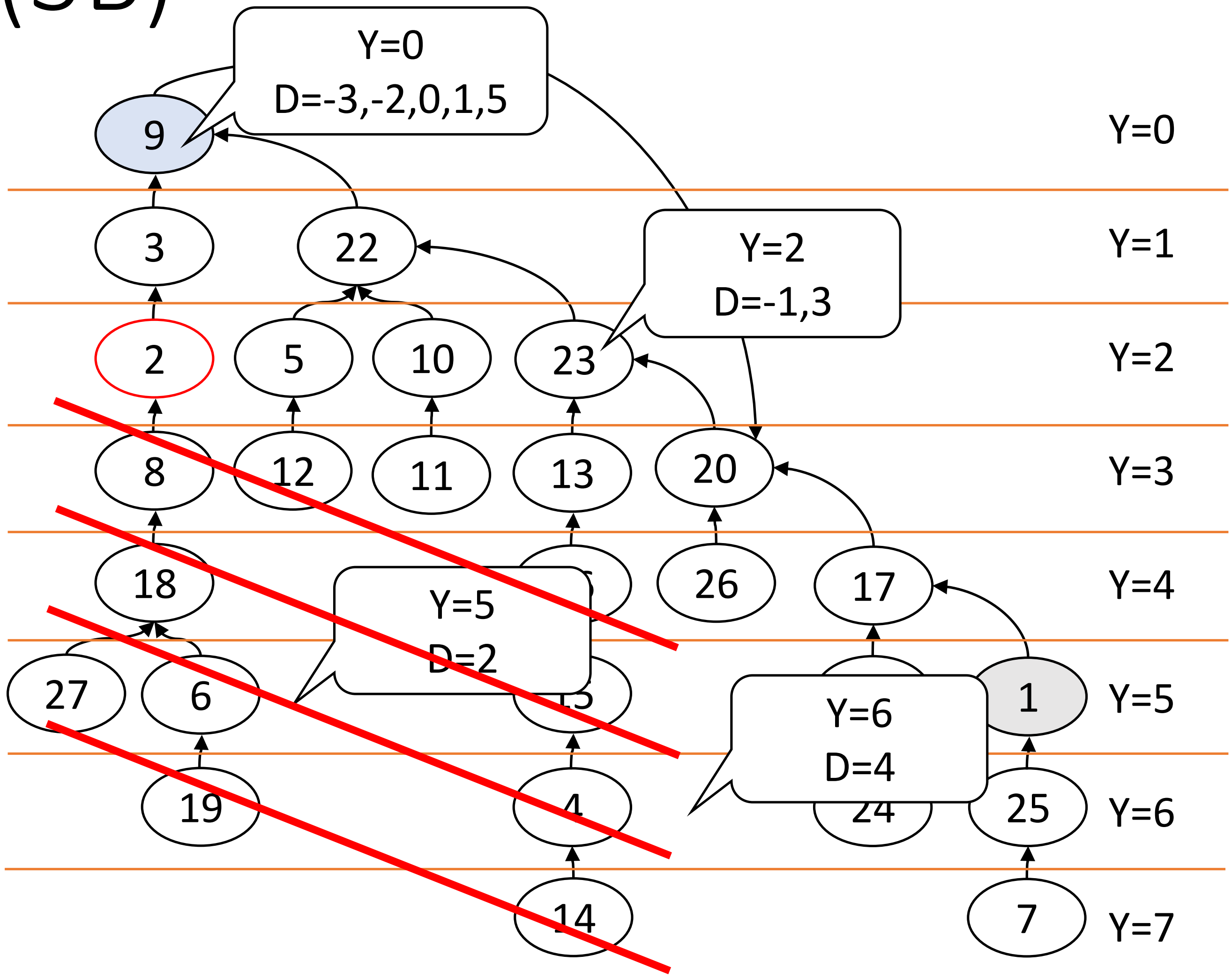
満点解法 – 場合分け(3B)

- 各イベントの境界部
  (赤線で示した)に注目する
- 意味があるものだけ処理したい
- 例えば、2を処理中のとき
  - 境界がこの範囲にあるイベントだけを処理すればよい
  - 実際には、さらにYの値が一致するイベントのみを対象にする



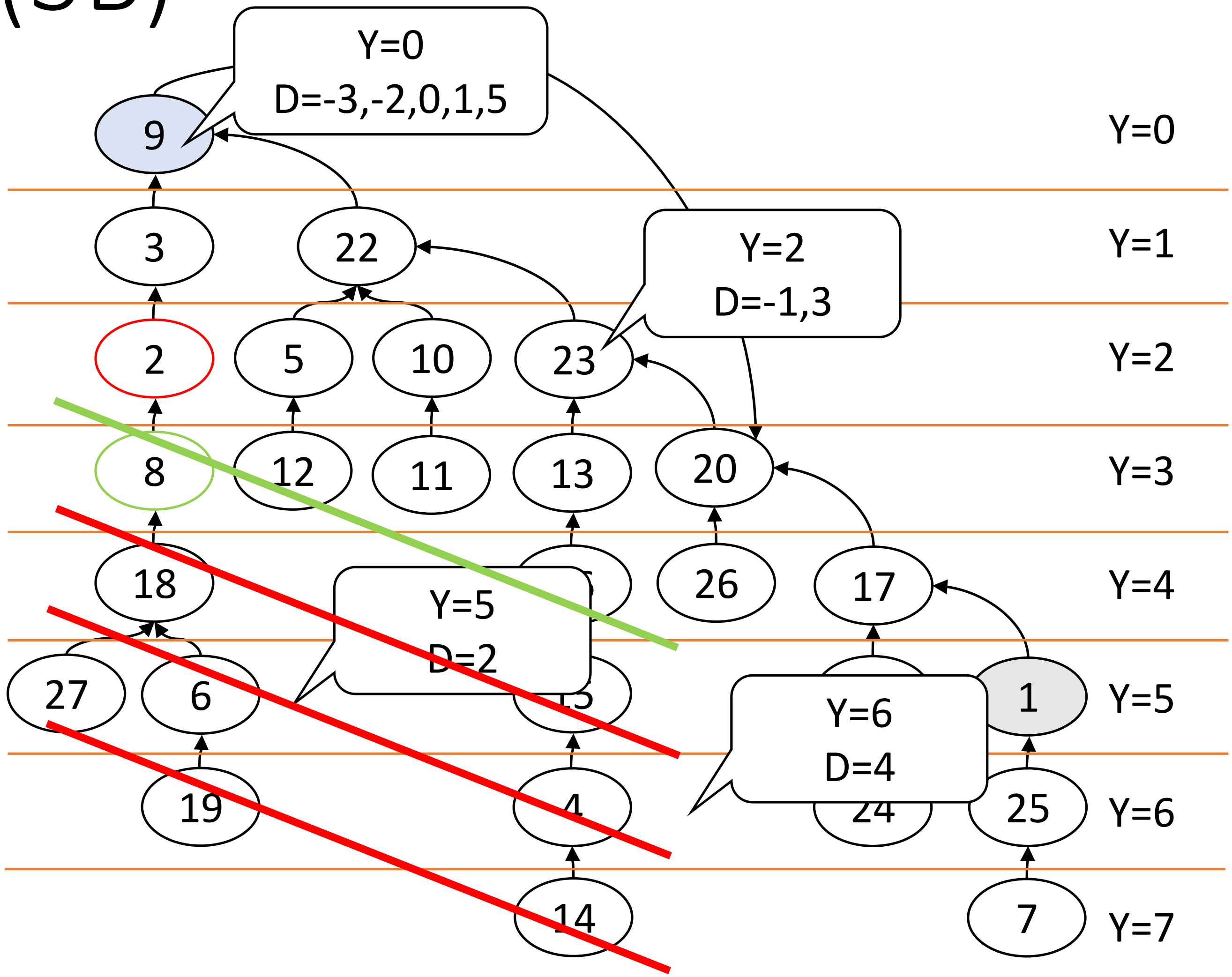
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける



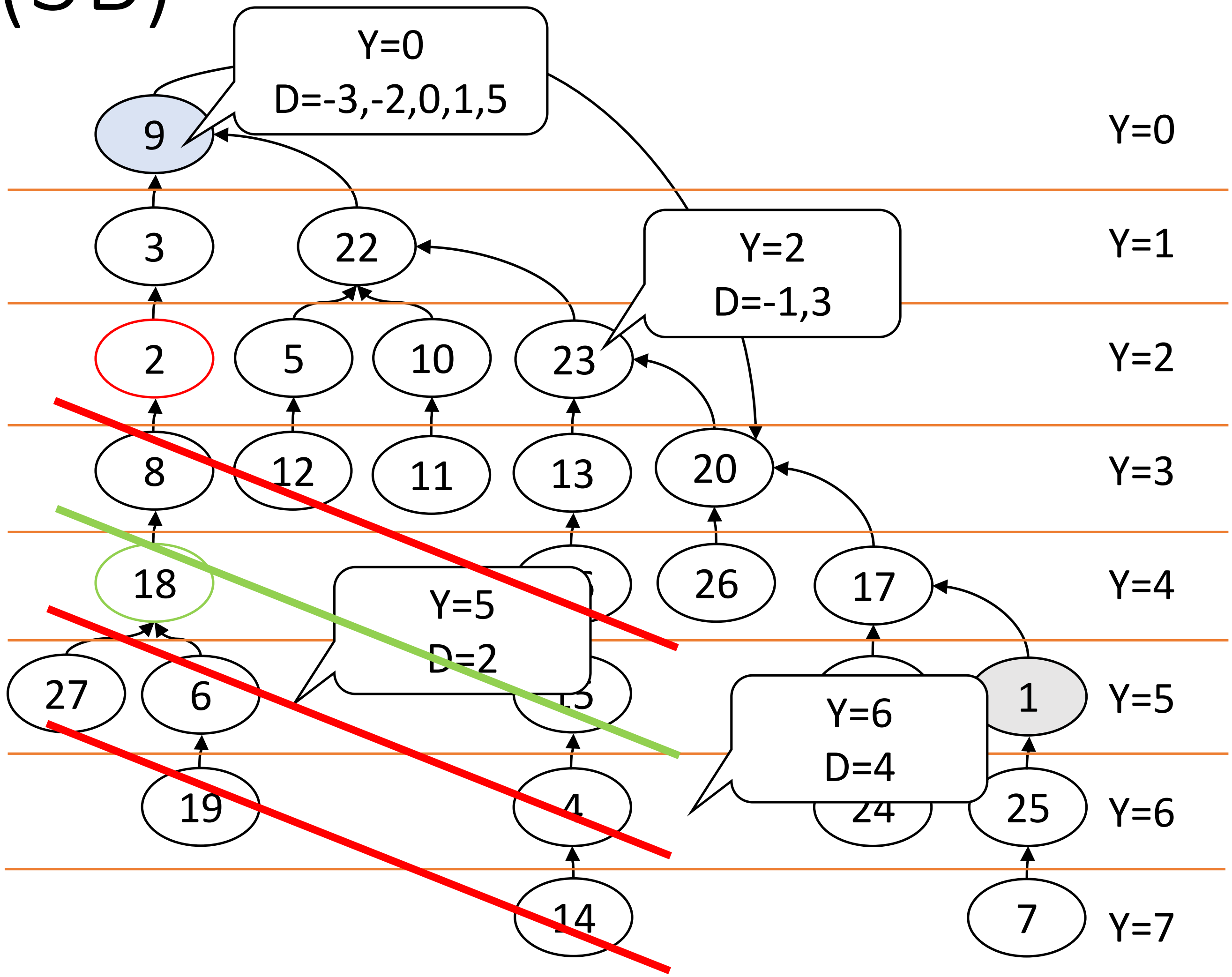
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける



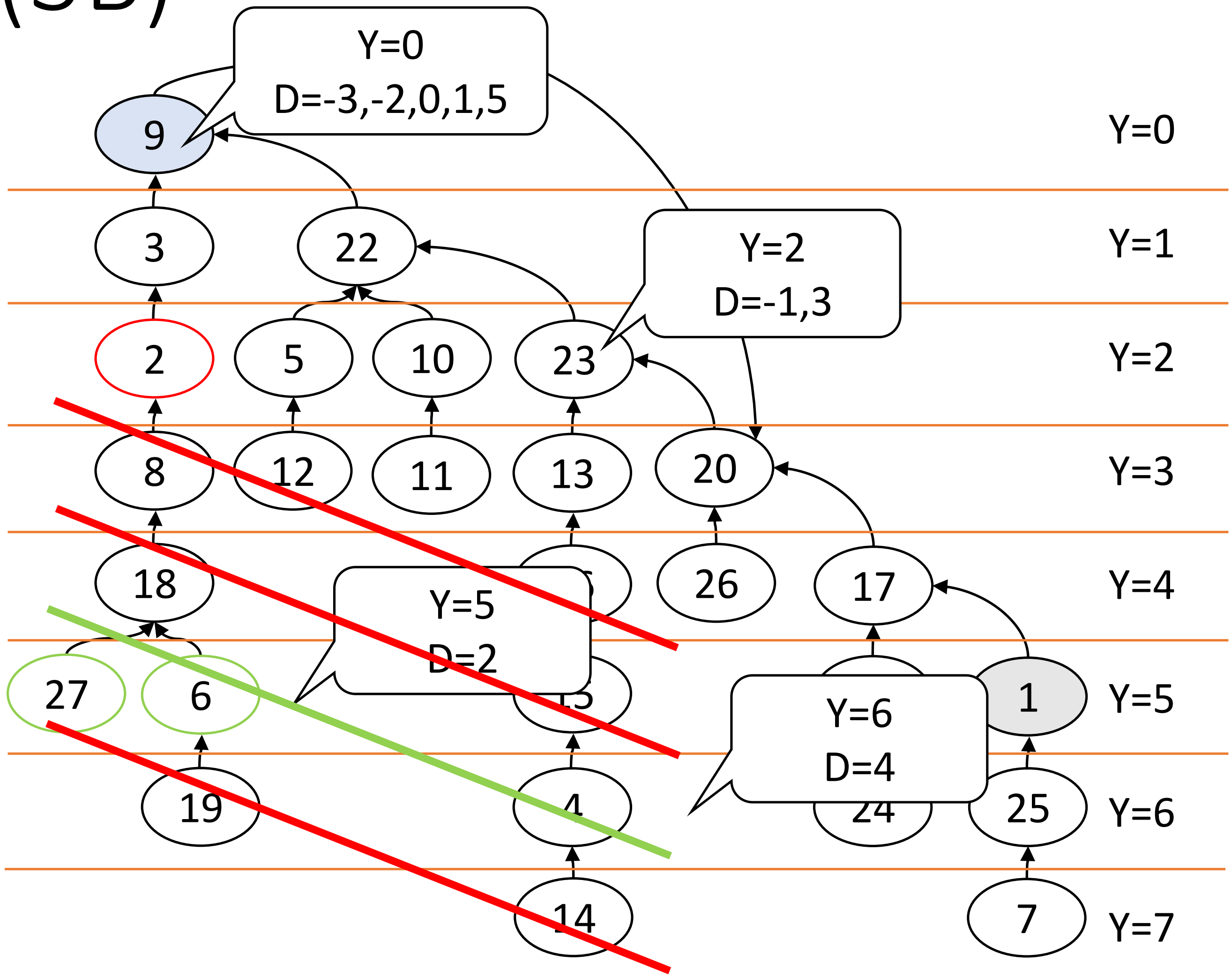
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける



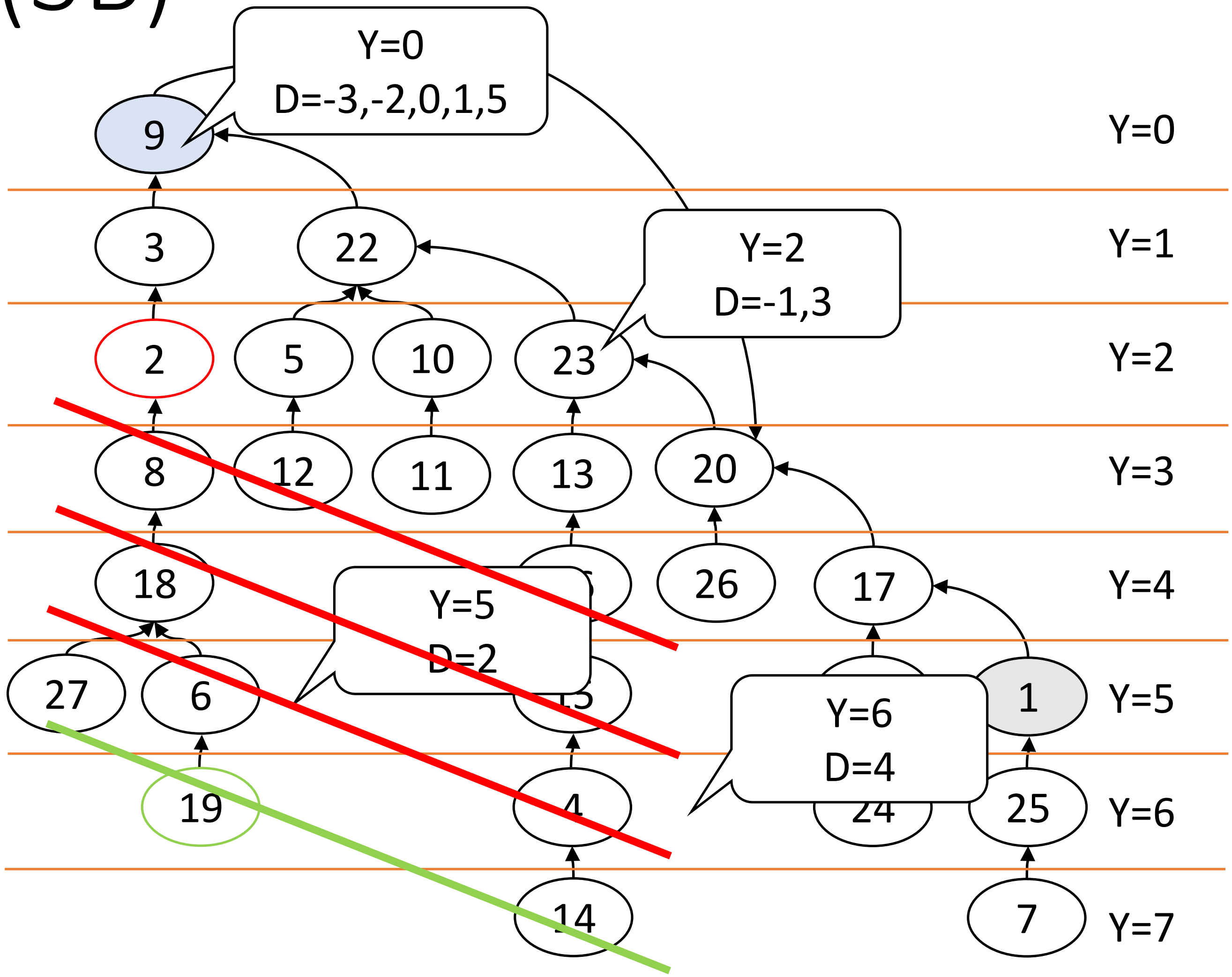
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける



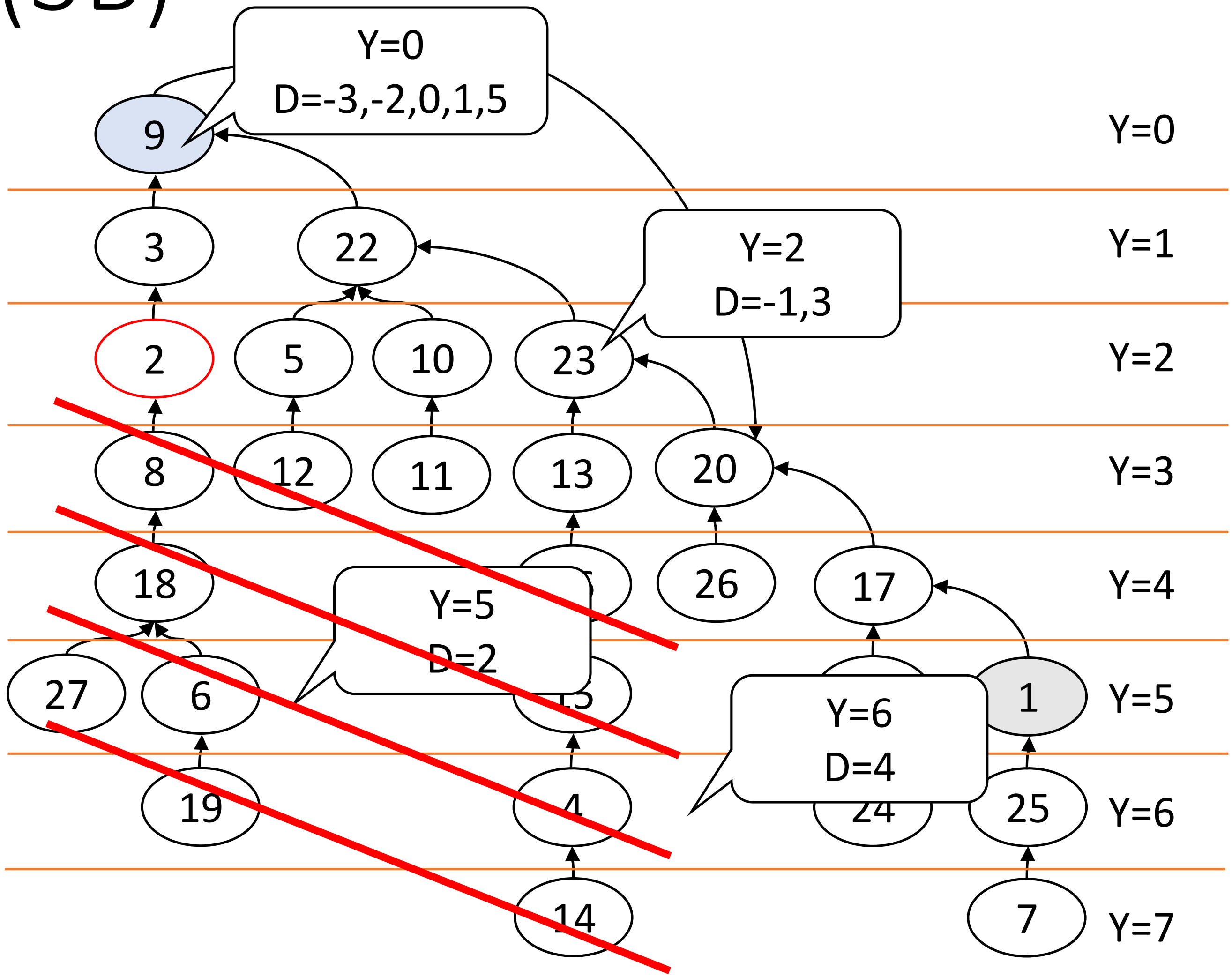
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける



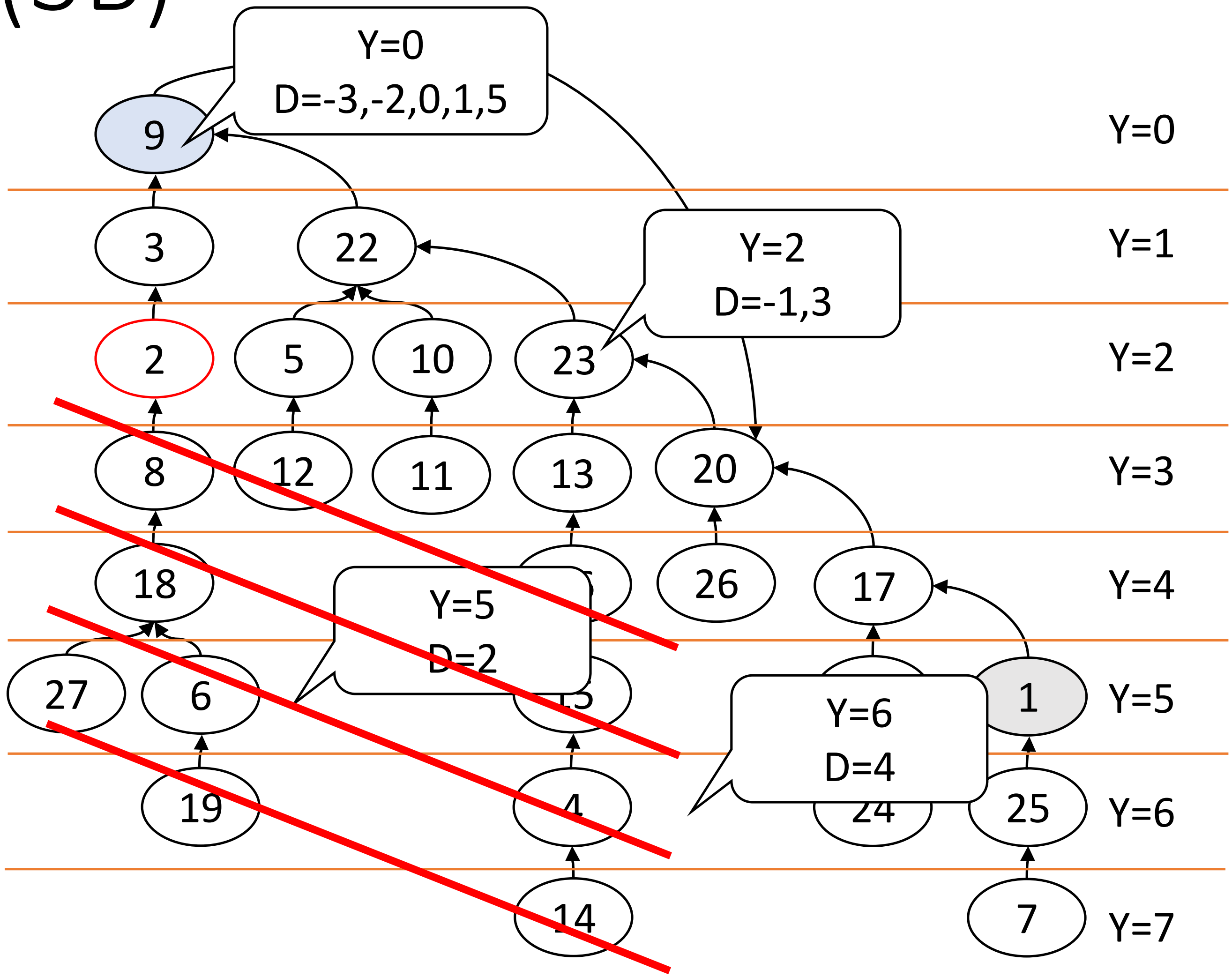
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける
  - 境界のすぐ下の頂点であって
  - 処理中の頂点の子孫であるようなもの



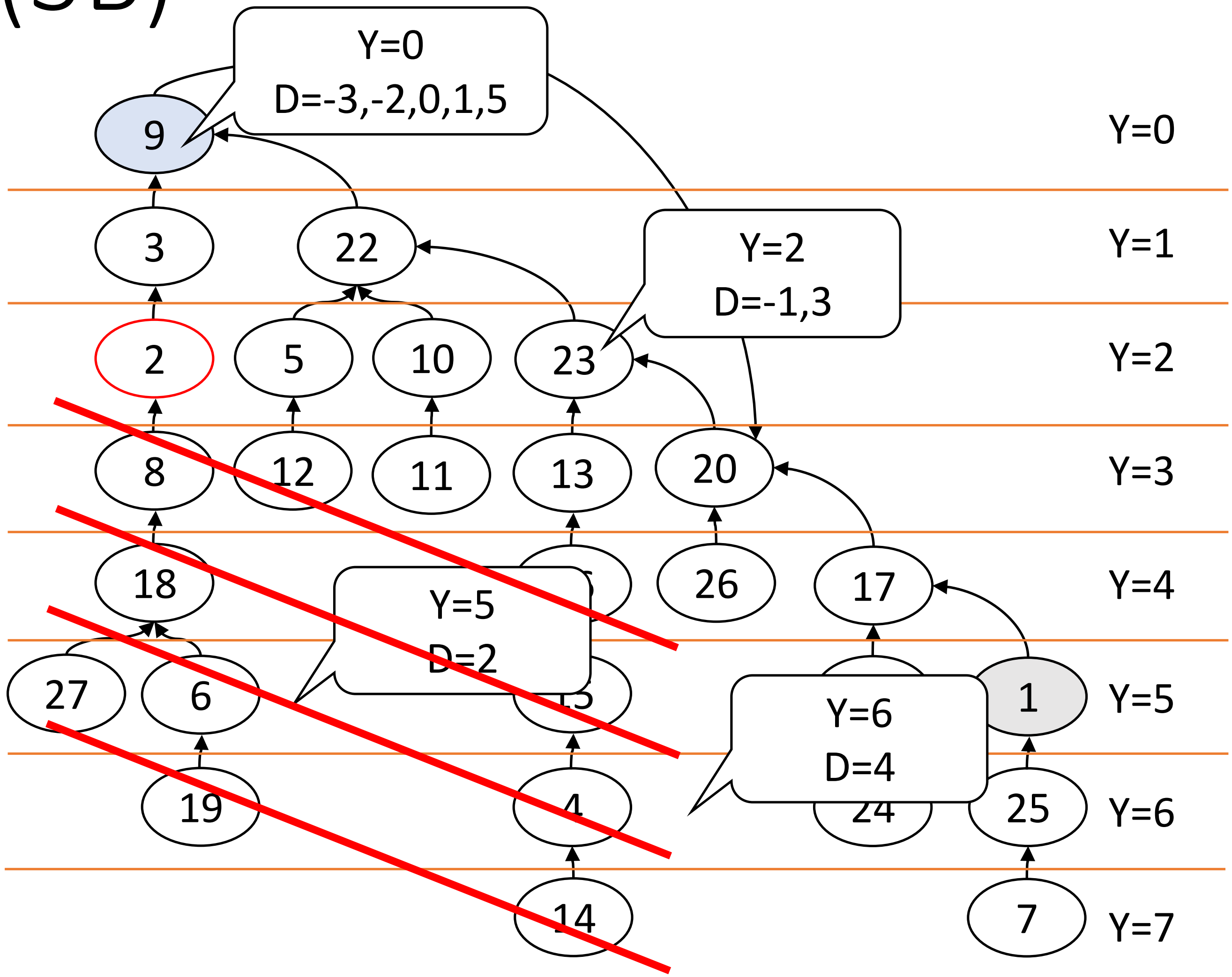
満点解法 – 場合分け(3B)

- 意味のあるイベントに、頂点を対応づける
  - 境界のすぐ下の頂点であって
  - 処理中の頂点の子孫であるようなもの
- 意味のあるイベントには、1つ以上の頂点が対応する



満点解法 – 場合分け(3B)

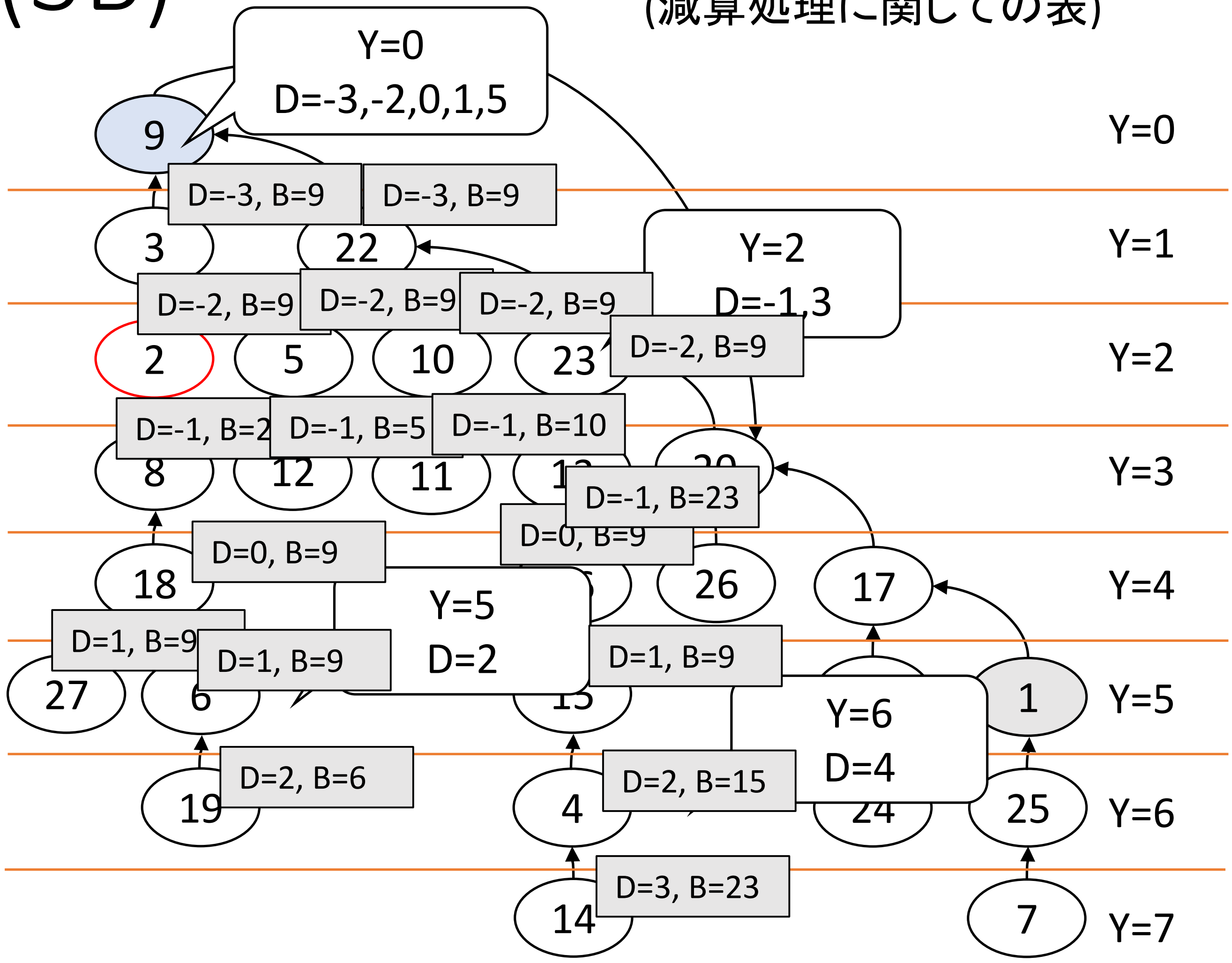
- 同じ頂点に対応するイベント呼び出しが2回行われることはない



満点解法 – 場合分け(3B)

各頂点に対応づけられたイベント。
(減算処理に関しての表)

- 同じ頂点に対応するイベント呼び出しが2回行われることはない



満点解法 – 場合分け(3B)

- 同じ頂点に対応するイベント呼び出しが2回行われることはない
- →意味のあるイベント呼び出しの回数は$O(n)$に抑えられた

満点解法 – 場合分け(3B)

- 例外的な場合の処理

満点解法 – 場合分け(3B)

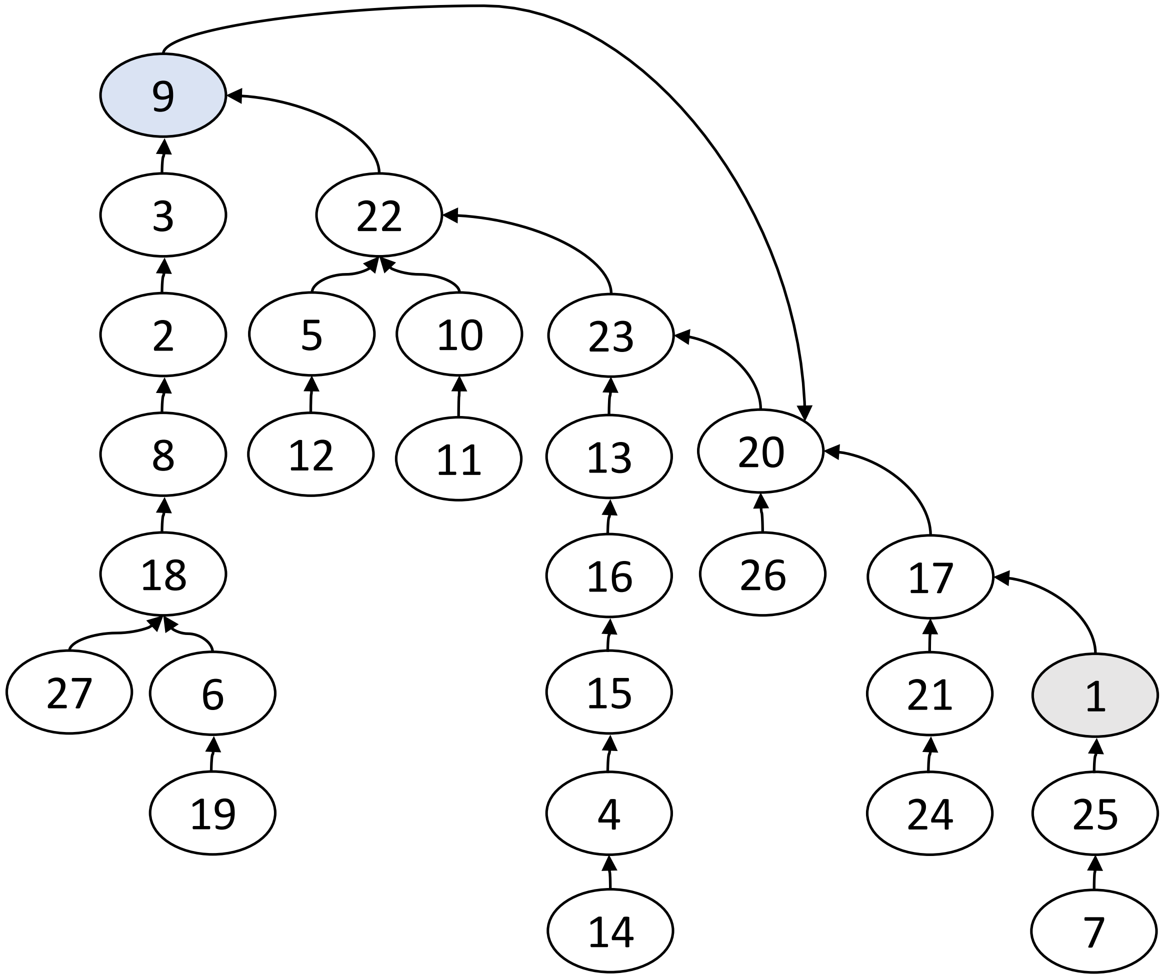
- 例外的な場合の処理
- 各イベント(Y,D)について、Y座標がY-(D+1)となるようなループ上の点が到達先の候補である。

満点解法 – 場合分け(3B)

- 例外的な場合の処理
- 各イベント(Y,D)について、Y座標がY-(D+1)となるようなループ上の点が到達先の候補である。
- Y-(D+1)がマイナスになる可能性があることだけ注意すればOK

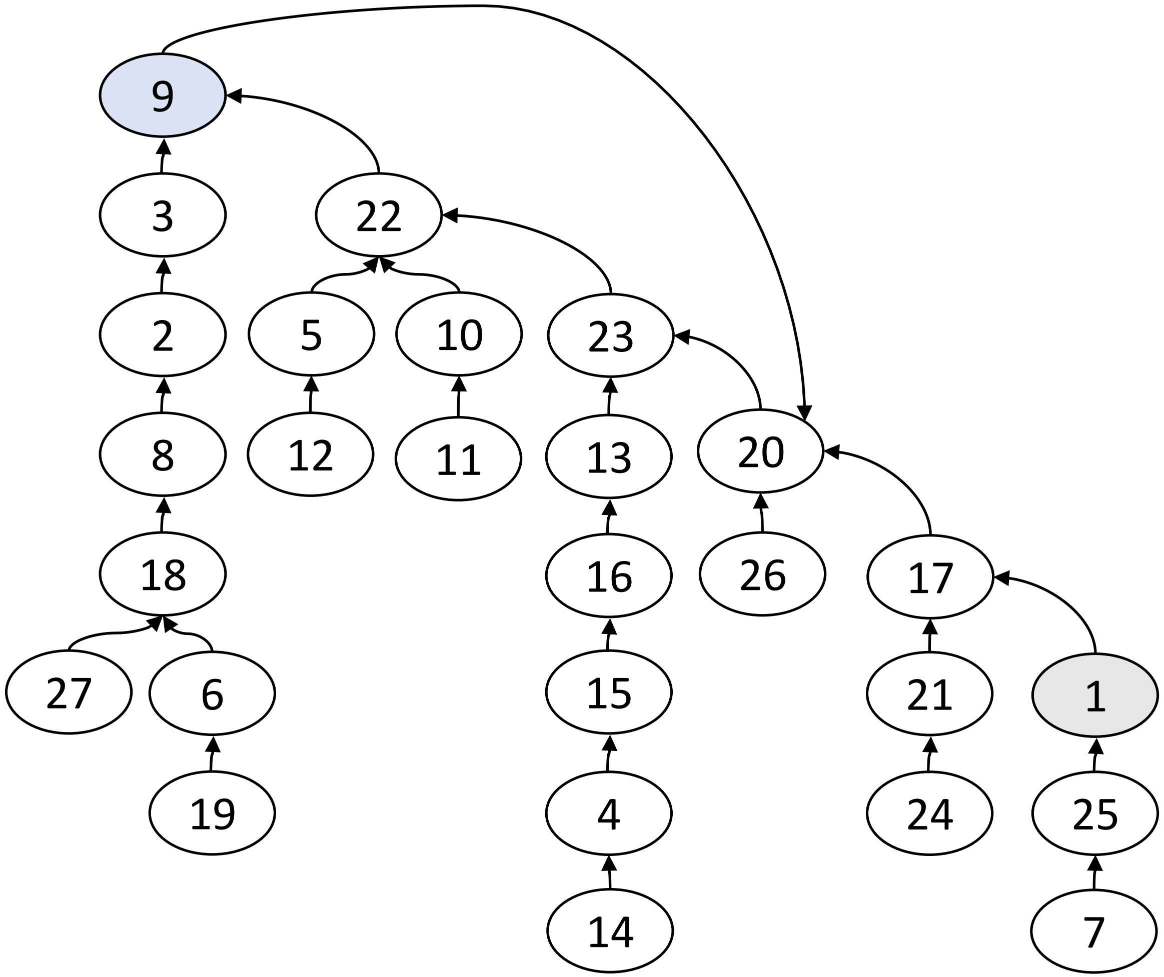
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える



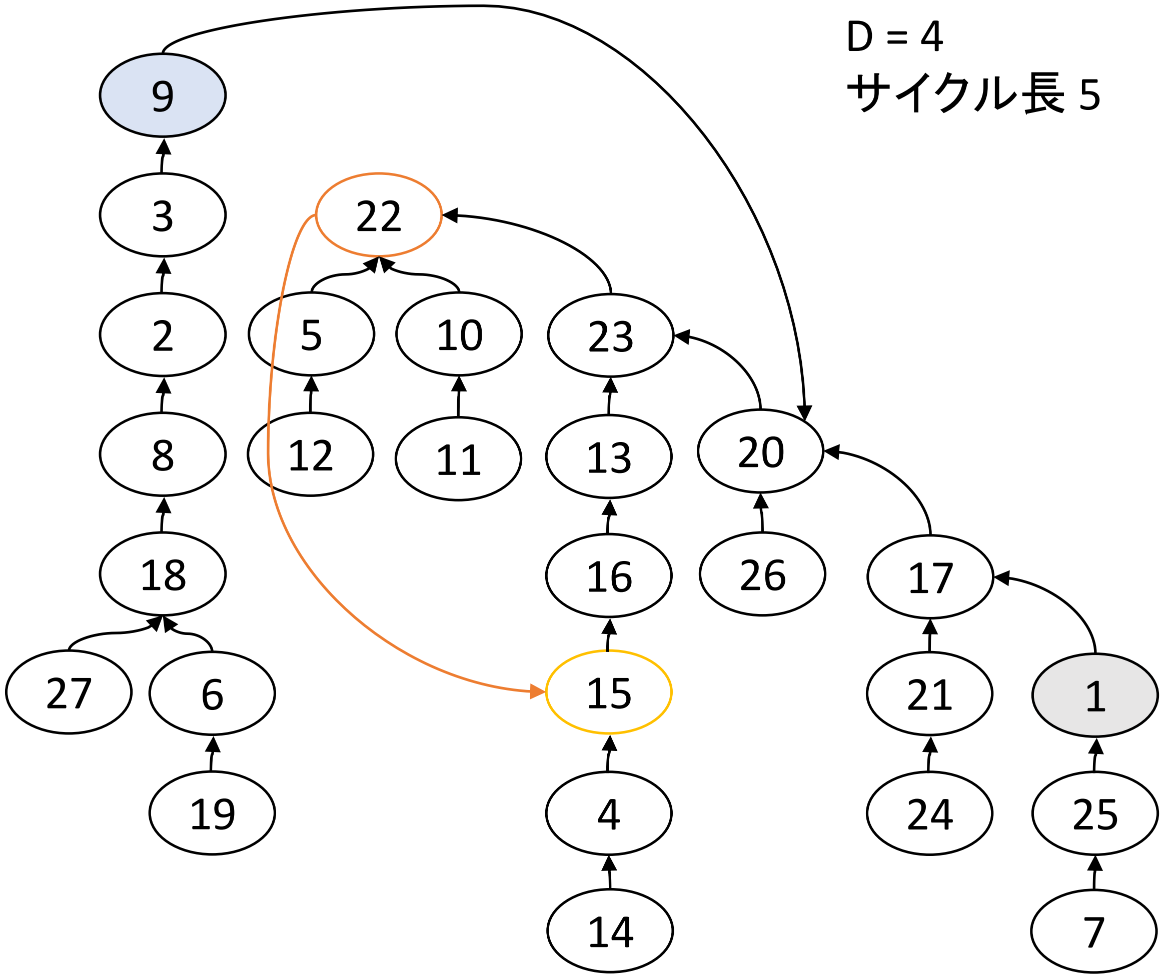
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える
- このとき、できるサイクル長は D + 1 である。



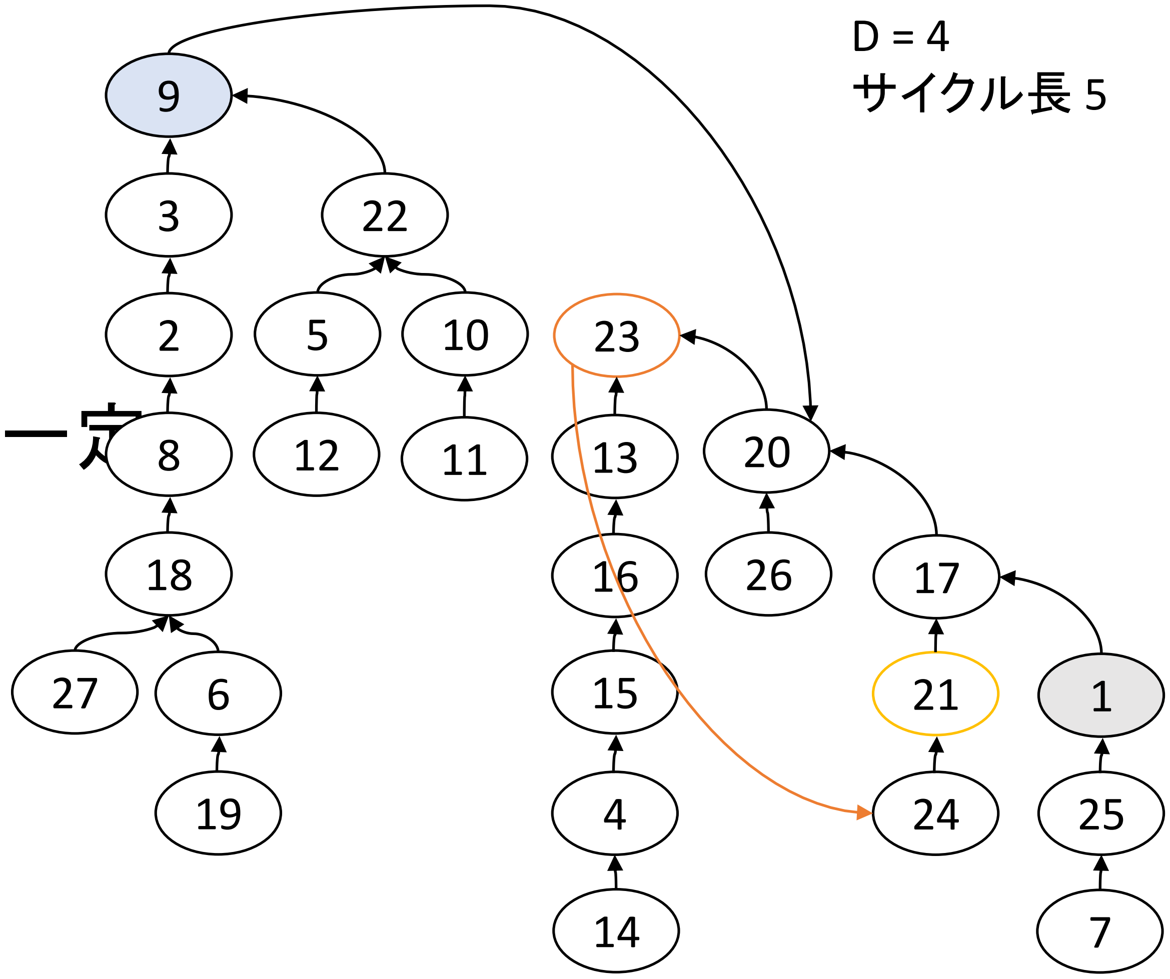
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える
- このとき、できるサイクル長は D + 1 である。



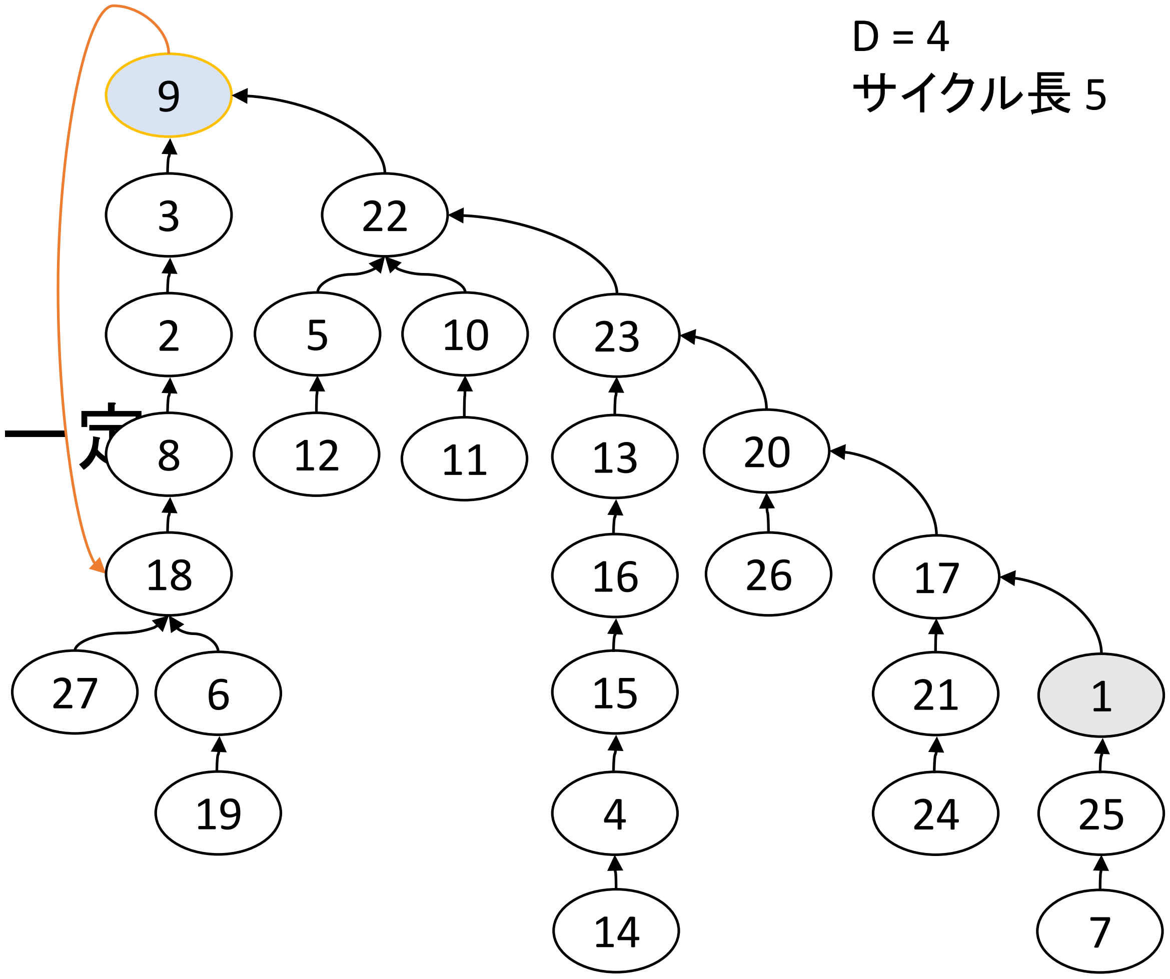
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える
- このとき、できるサイクル長は D + 1 である。
- D一定のとき、到達先のY座標は一定



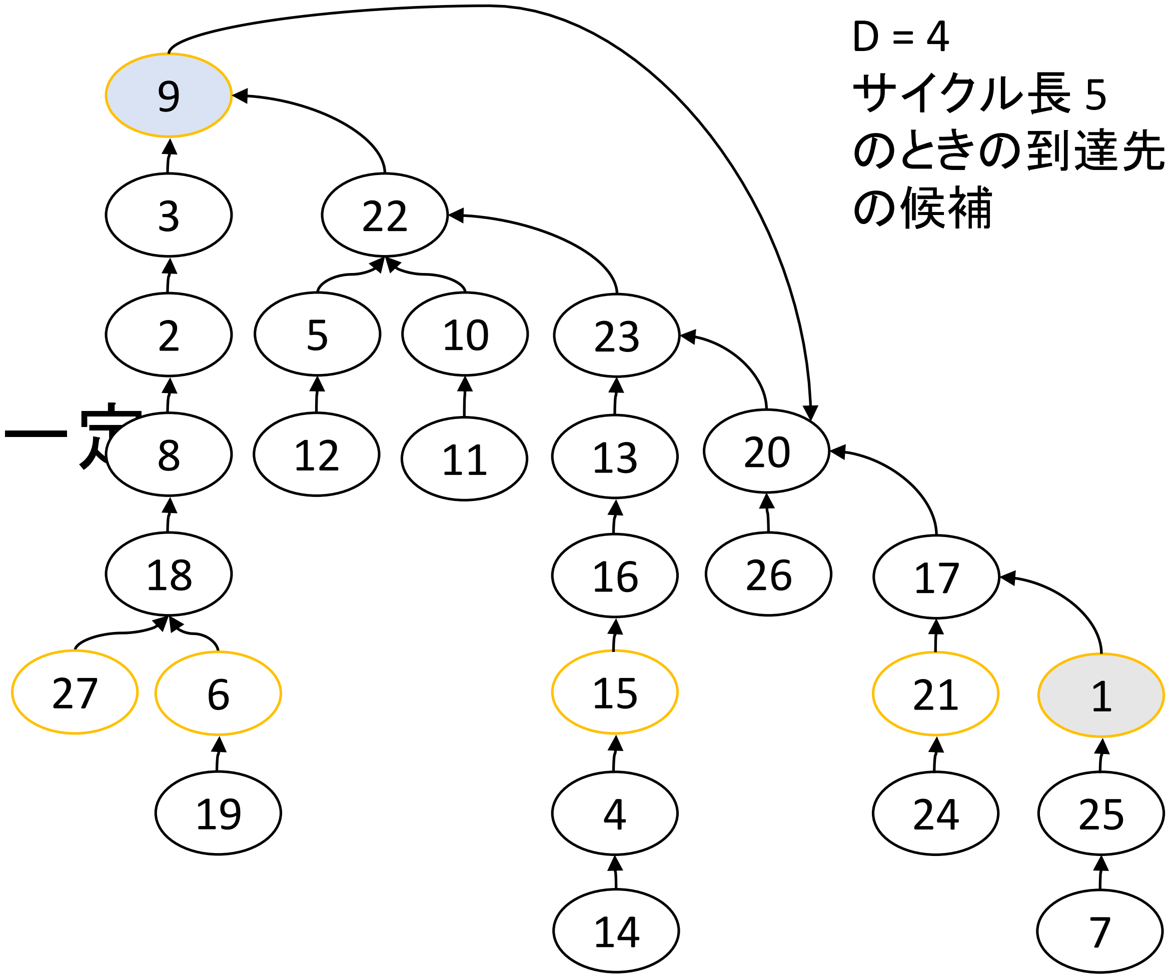
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える
- このとき、できるサイクル長は D + 1 である。
- D一定のとき、到達先のY座標は一定
- というわけではないが、D+1を法として一定である。



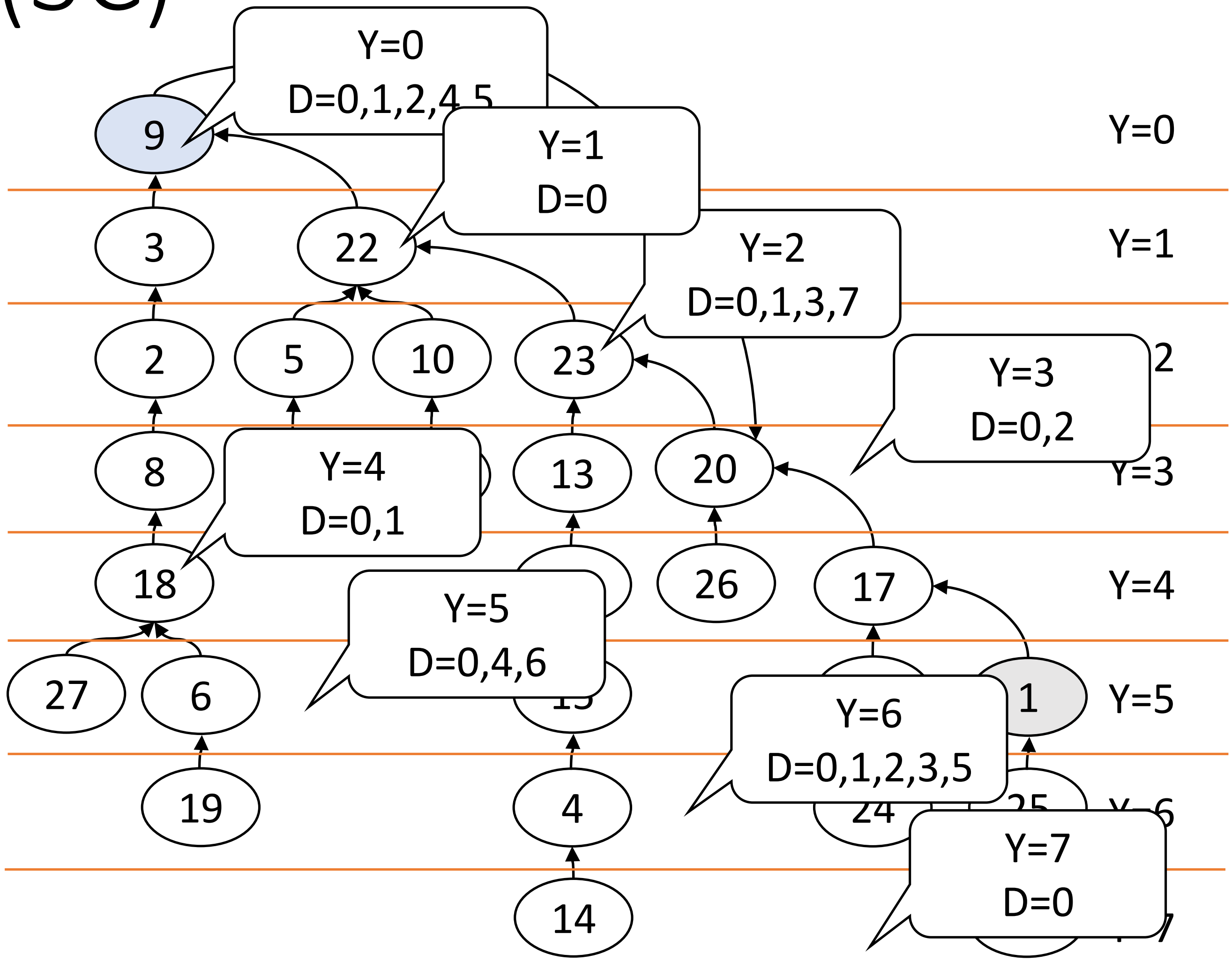
満点解法 – 場合分け(3C)

- AがBの祖先である場合を考える
- このとき、できるサイクル長は D + 1 である。
- D一定のとき、到達先のY座標は一定
- というわけではないが、D+1を法として一定である。



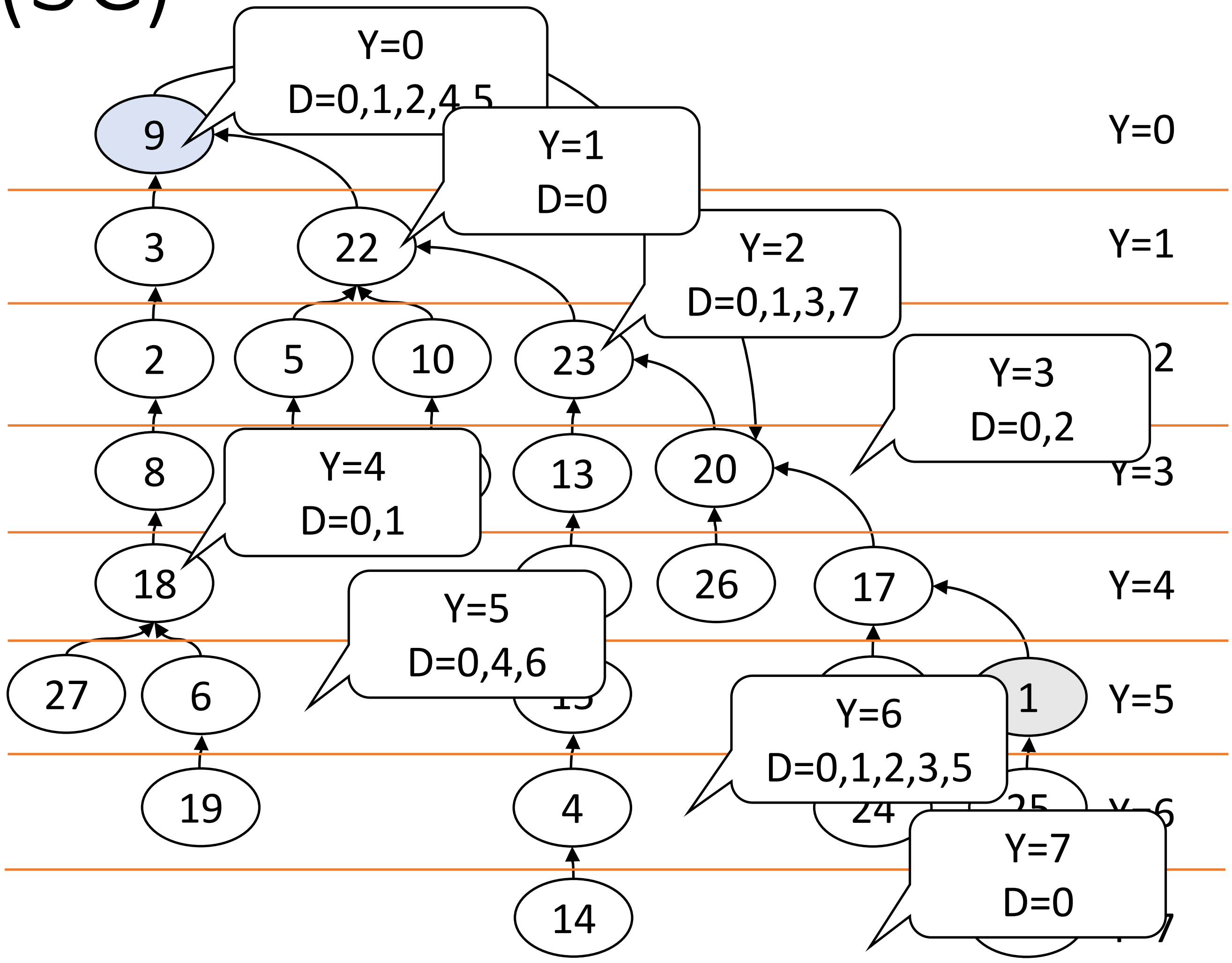
満点解法 – 場合分け(3C)

- 以上をイベントとして管理



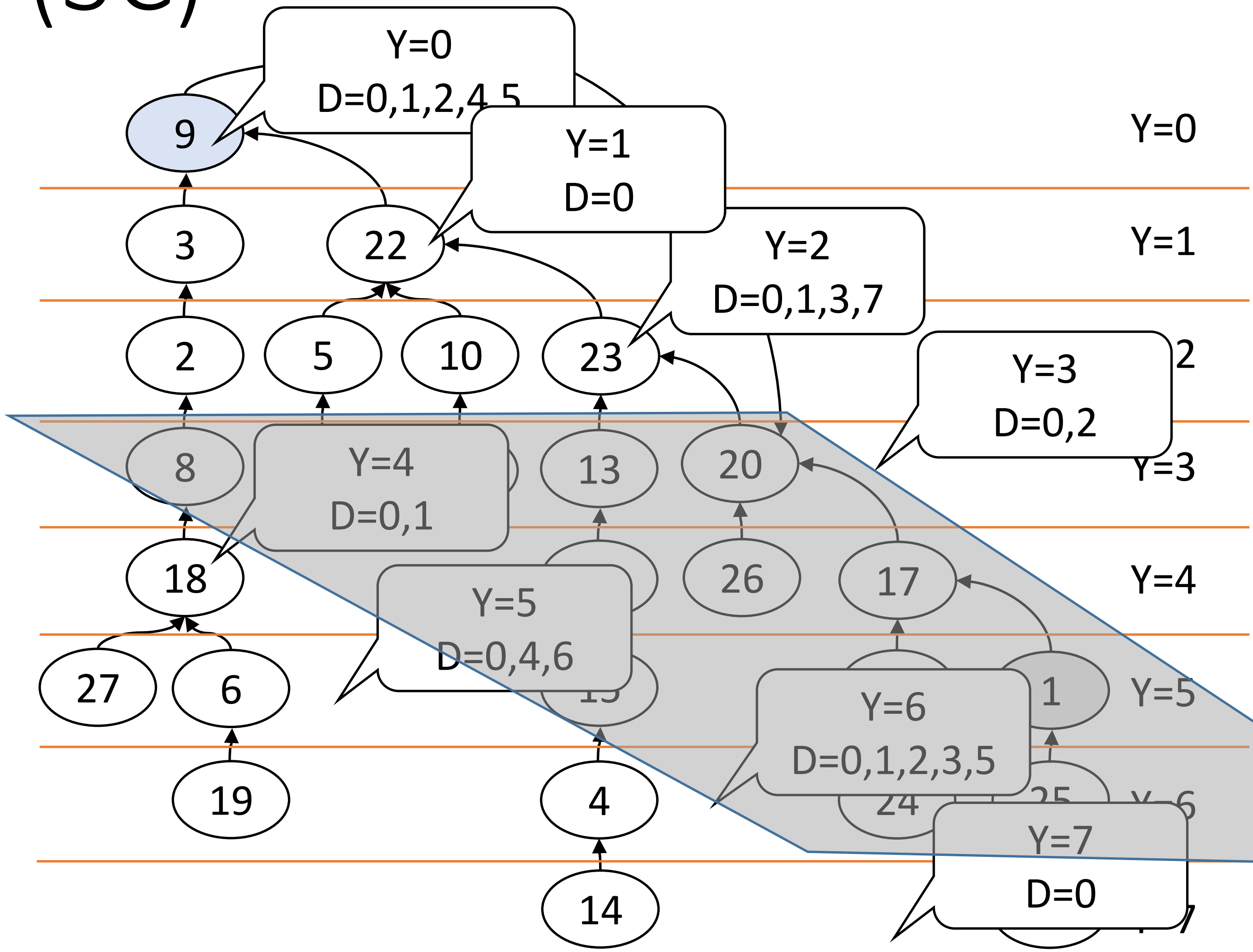
満点解法 – 場合分け(3C)

- 以上をイベントとして管理
- イベントの個数は逆数の和だから、$O(n \log n)$



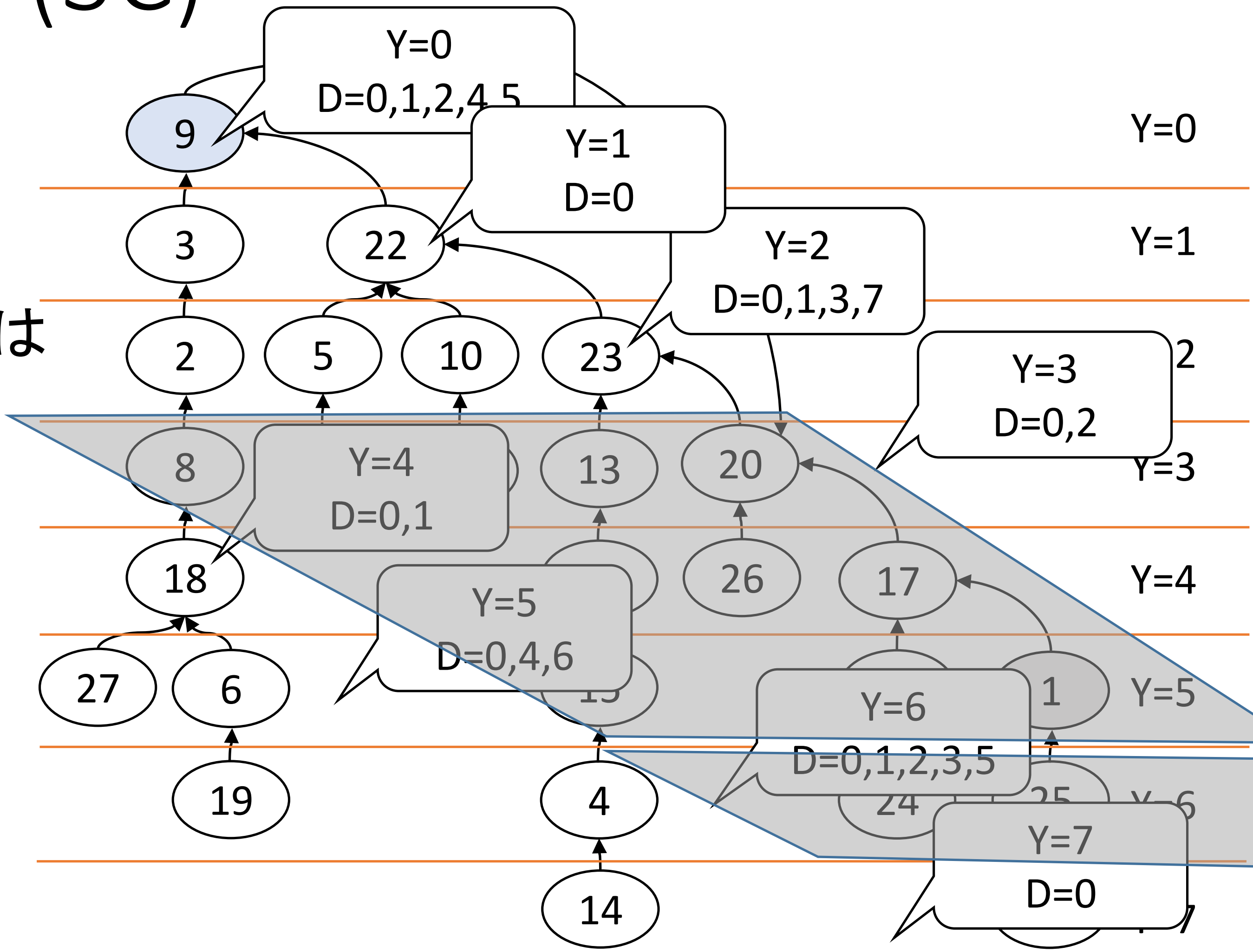
満点解法 – 場合分け(3C)

- D=3のイベントに引っかかるBの範囲は図のようになる



満点解法 – 場合分け(3C)

- D=3のイベントに引っかかるBの範囲は図のようになる
  - そのうち、Y=2, Y=6のものの範囲はそれぞれ図のようになる



満点解法 – 場合分け(3C)

- あとは(3B)の場合と同様に、木を下から走査する

満点解法 – 場合分け(3C)

- あとは(3B)の場合と同様に、木を下から走査する
  - 木を走査しながら、頂点ごとに子孫の個数を記録したFenwick木を構築
  - 各頂点について、意味のあるイベントを列挙し、それらのイベントについて数え上げを行う
  - 各頂点について、意味のないイベントは一括して数え上げを行う

満点解法 – 場合分け(3C)

- これによって、同様の計算量解析が使える
- 到達先が尾根(Y=X)にないときの処理の計算量は合わせて $O(n \log n)$になることが示せる

満点解法 – 場合分け(3C)

- これによって、同様の計算量解析が使える
- 到達先が尾根(Y=X)にないときの処理の計算量は合わせて $O(n \log n)$になることが示せる
- 到達先が尾根にあるときの処理→一回あたり$O(\log n)$の処理を合計$O(n \log n)$回行うので、$O(n \log^2 n)$までは抑えられる

満点解法

- 以上により時間計算量$O(n \log^2 n)$を達成できた