アルゴリズム入門(6)
(動的計画法)

宮崎修一
京都大学　学術情報メディアセンター

動的計画法の考え方：

元の問題から、よりサイズの小さい「部分問題」を定義する。

部分問題を解いた結果を表にまとめておき、
より大きなサイズの部分問題を解くときに、表の結果を利用する。

小さい部分問題から順番に解いて行って、最後の部分問題が
元の問題そのものである。（よって、最後には元の問題が
解けたことになる。）

行列の掛け算

２×３行列　　３×４行列

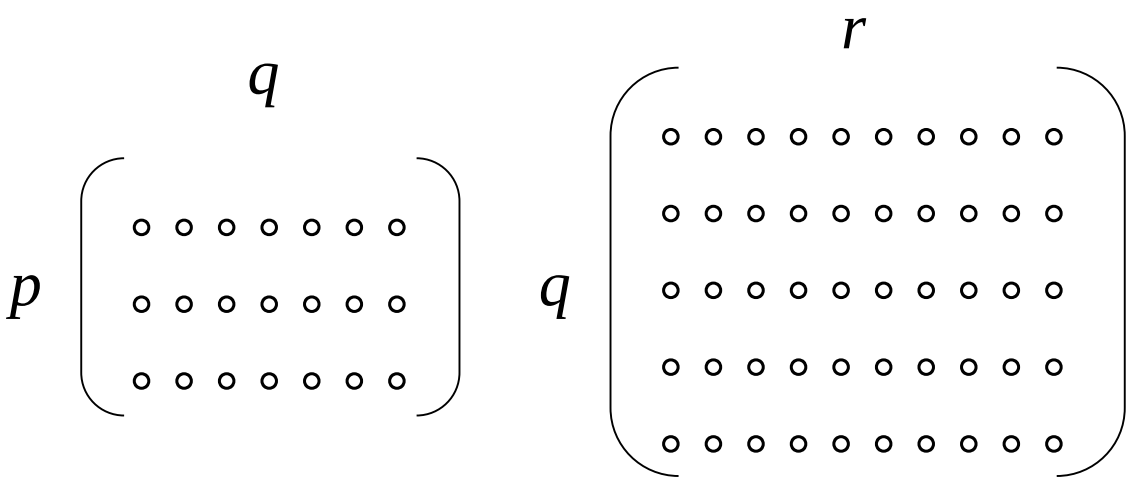
$$\begin{pmatrix} 3 & 6 & 1 \\ 4 & 0 & 2 \end{pmatrix} \begin{pmatrix} 3 & 8 & 1 & 6 \\ 1 & 0 & 1 & 2 \\ 2 & 2 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 17 & 26 & 9 & 30 \\ 16 & 36 & 4 & 24 \end{pmatrix}$$

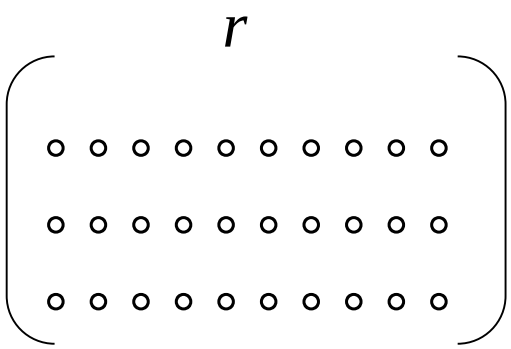
２×４行列

問題：掛け算は
全部で何回？

$(p \times q)$行列　　$(q \times r)$行列



$=$



$(p \times r)$行列

掛け算の回数は全体で
$p \times q \times r$回

4つの行列の掛け算

$A_1 \times A_2 \times A_3 \times A_4$

（50×2）　（2×80）　（80×3）　（3×20）

・掛け算結果は、順番に依らない。
・掛け算回数は、順番に大きく依存する。

● $A_1 \times A_2 \rightarrow 50 \times 2 \times 80 =$ 8000回　結果は50×80行列

● $A_3 \times A_4 \rightarrow 80 \times 3 \times 20 =$ 4800回　結果は80×20行列

● × ● → 50×80×20＝80000回

計　92800回

● $A_2 \times A_3 \rightarrow 2 \times 80 \times 3 =$ 480回　結果は 2× 3行列

● $A_1 \times$ ● → 50× 2× 3＝ 300回　結果は50× 3行列

● × $A_4 \rightarrow 50 \times 3 \times 20 =$ 3000回

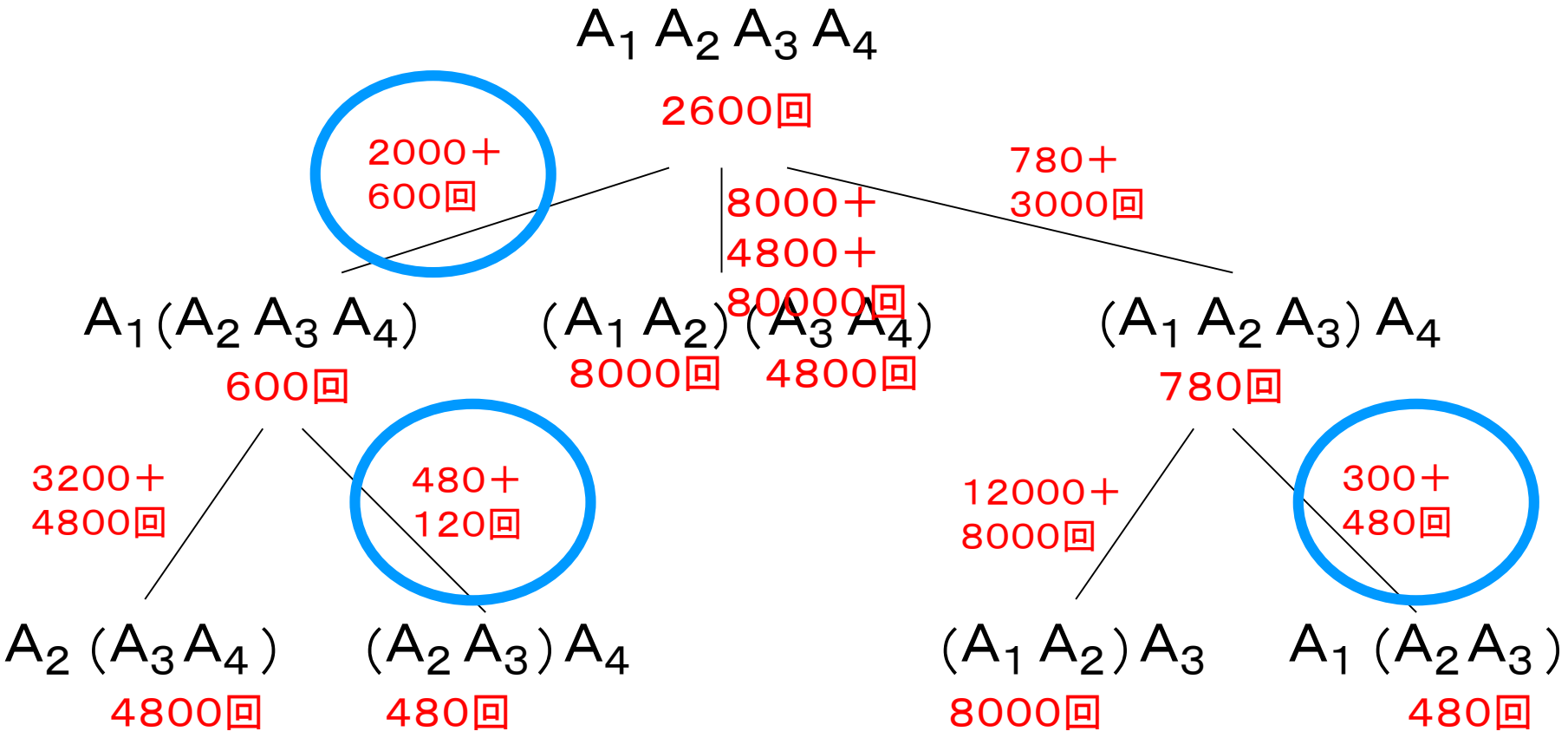
計　3780回

問題：これより少ない回数で出来る？

n個の行列が与えられたとき

$$A_1 \times A_2 \times \cdots \times A_n$$

どういう順番で掛けていくのが、掛け算回数が最も少ないか？



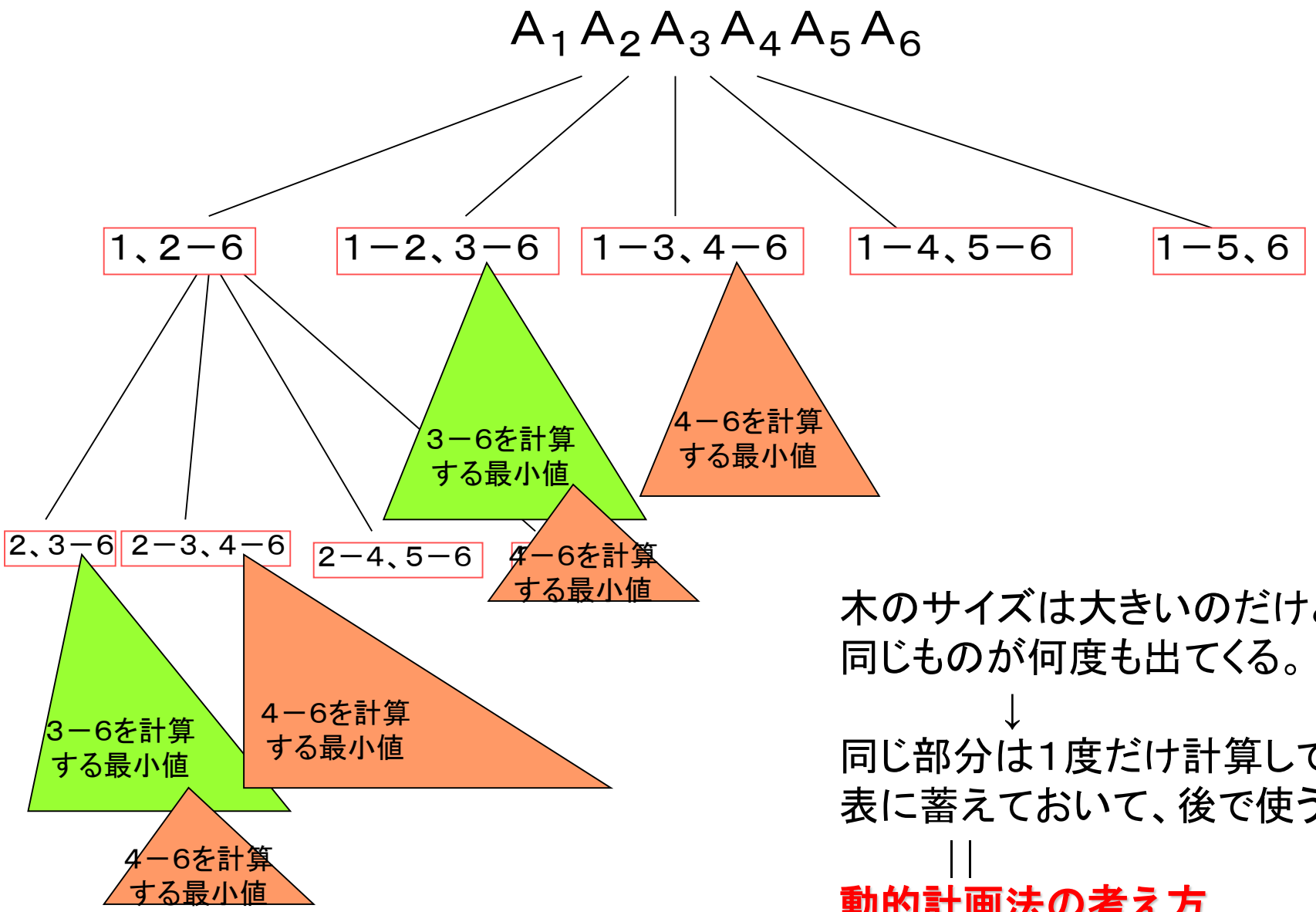
n個の場合、場合分けは$\geqq 4^n$

木全体を計算していたのでは、多項式時間では収まらない。

↓

動的計画法

動的計画法の考え方（n＝6の場合の例）



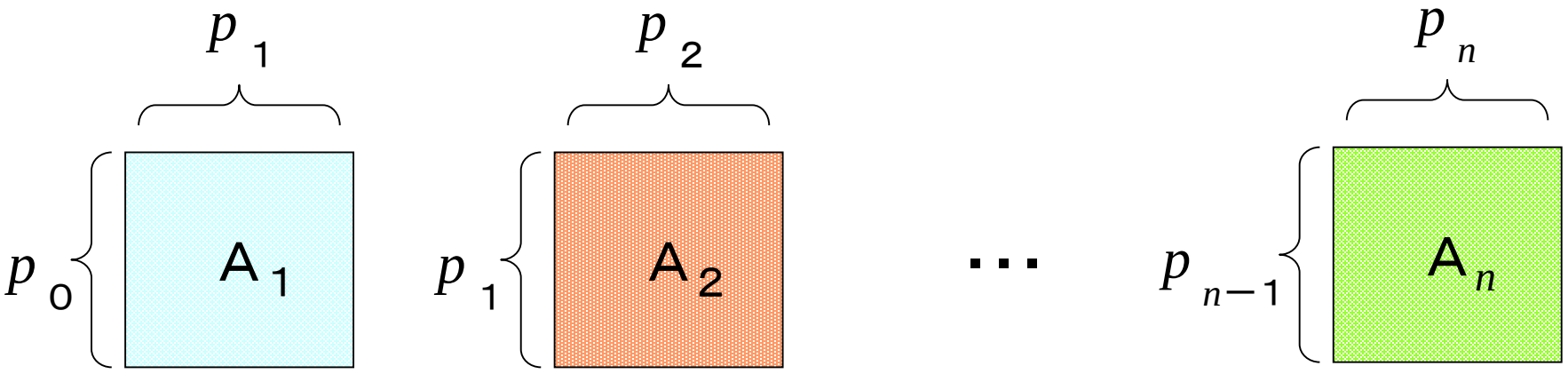
木のサイズは大きいのだけど、
同じものが何度も出てくる。

↓

同じ部分は1度だけ計算して
表に蓄えておいて、後で使う。

||

動的計画法の考え方



$m[i, j]$: $A_i\ A_{i+1}$ ・・・ $A_{j-1}\ A_j$ を計算するのに、掛け算回数を一番少なくした場合の掛け算回数。

つまり、求めたいのは $m[1, n]$。

第1ステップ：

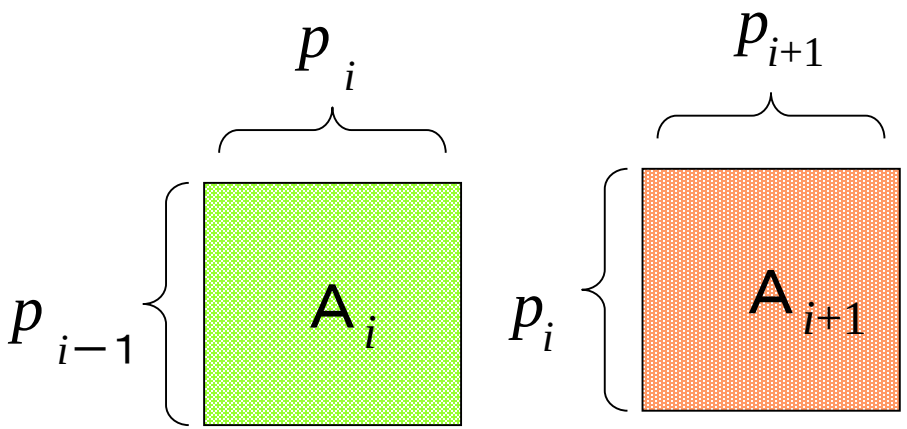
$m[i, i]=0$（全てのiに対して）

つまり、A_i を求めるのに掛け算回数は0回。

第2ステップ：

$m[i, i+1]$ を求める（$1 \leqq i \leqq n-1$）

つまり、A_i A_{i+1}を求める掛け算回数。



$m[i, i+1]= p_{i-1}\, p_i\, p_{i+1}$

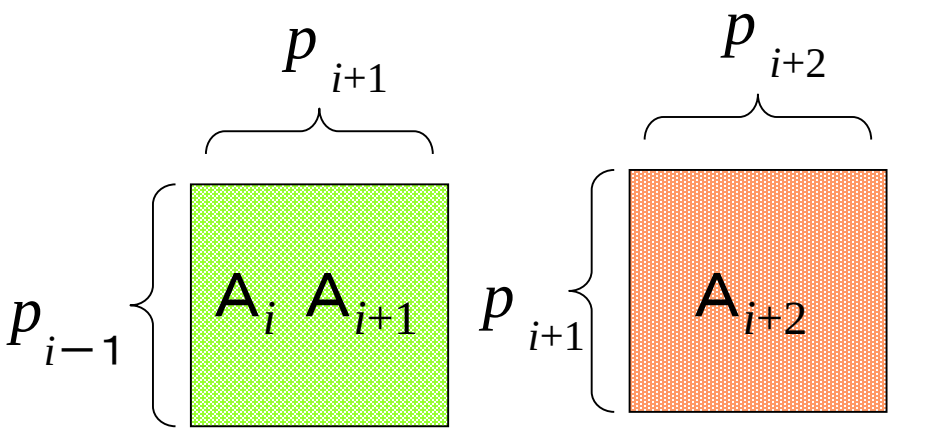
第3ステップ：

$m[i, i+2]$ を求める（$1 \leqq i \leqq n-2$）

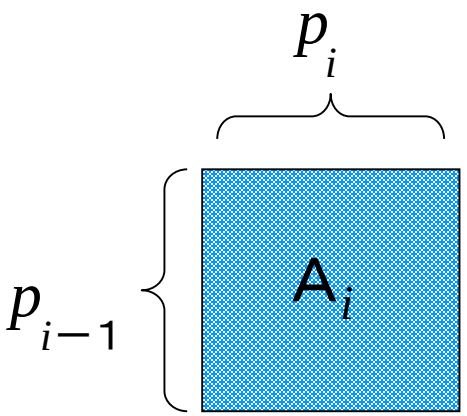
つまり、$A_i\ A_{i+1} A_{i+2}$ を求める掛け算回数。

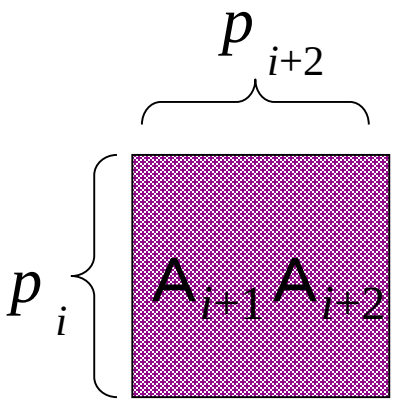
この2通りのうち、少ない方。

$(A_i\ A_{i+1})A_{i+2}$



$A_i(A_{i+1} A_{i+2})$





$$m[i, i+2] = \min\{m[i, i+1] + m[i+2, i+2] + p_{i-1}\ p_{i+1}\ p_{i+2}\ ,$$
$$m[i, i] + m[i+1, i+2]\} + p_{i-1}\ p_i\ p_{i+2}\}$$

第kステップ：

$m[i, i+k-1]$ を求める（$1 \leqq i \leqq n-k+1$）

$$A_i\ A_{i+1} A_{i+2}\ \cdots A_{i+k-2} A_{i+k-1}$$

$$A_i(A_{i+1} A_{i+2}\ \cdots A_{i+k-2} A_{i+k-1}\ )$$

$$(A_i\ A_{i+1})\ (A_{i+2}\ \cdots A_{i+k-2} A_{i+k-1}\ )$$

$$\vdots$$

$$(A_i\ A_{i+1} A_{i+2} \cdots A_{i+k-3})(A_{i+k-2} A_{i+k-1}\ )$$

$$(A_i\ A_{i+1} A_{i+2}\ \cdots A_{i+k-2}\ )A_{i+k-1}$$

$$\begin{aligned} m[i, i+k-1] = \min\{ & m[i, i]+ m[i, i+k-1]+ p_{i-1}\ p_i\ p_{i+k-1}\ , \\ & m[i, i+1]+m[i+2, i+k-1]+ p_{i-1} p_{i+1}\ p_{i+k-1}\ , \\ & \cdots \\ & m[i, i+k-3]+m[i+k-2,i+k-1]+p_{i-1}\ p_{i+k-3}\ p_{i+k-1}\ , \\ & m[i, i+k-2]+m[i+k-1,i+k-1] + p_{i-1} p_{i+k-2}\ p_{i+k-1}\ \} \end{aligned}$$

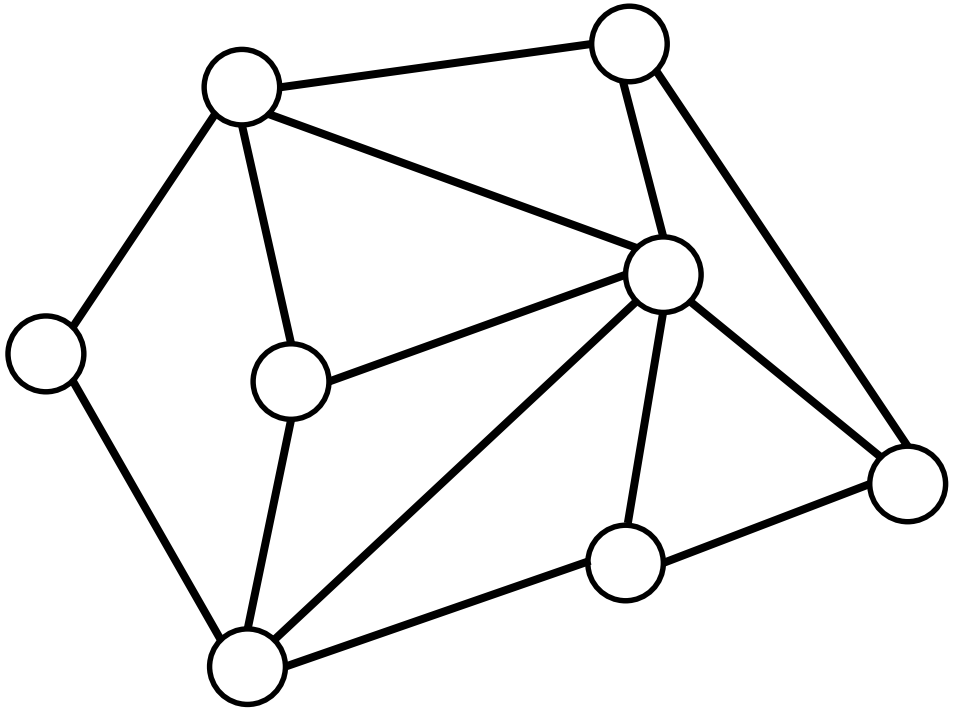
同様に進めて、第nステップで$m[1, n]$が求まる。

計算時間は$O(n^3)$。

$m[i, j]$は$O(n^2)$。
1つを計算するのに$O(n)$。

最大独立頂点集合問題（以前やった）

入力：グラフ $G=(V, E)$

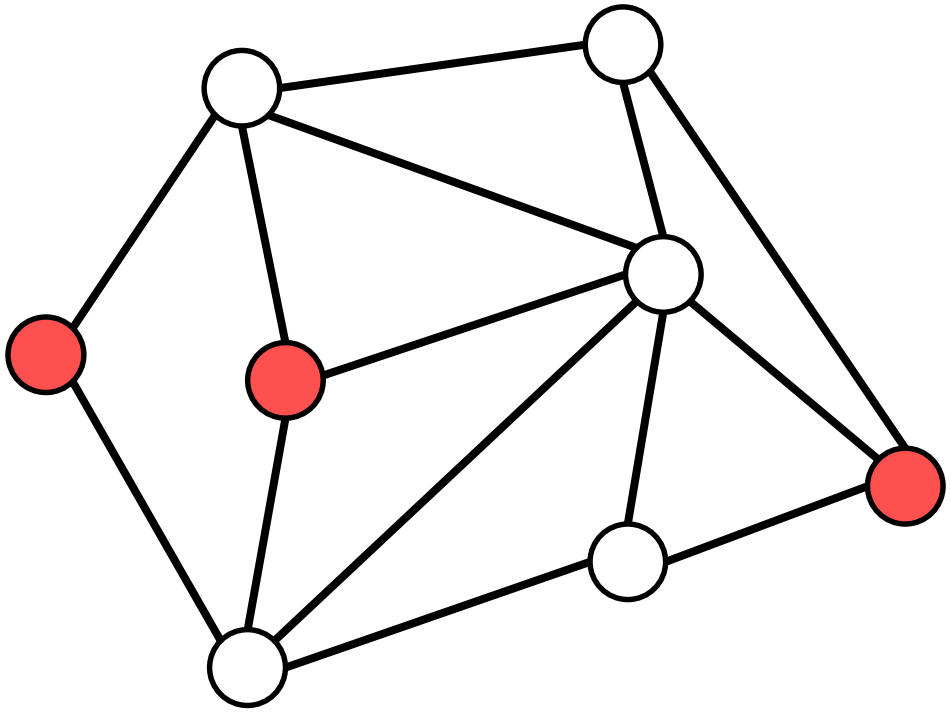


独立頂点集合：間に枝のない頂点集合

最大独立頂点集合問題（以前やった）

入力：グラフ $G=(V, E)$

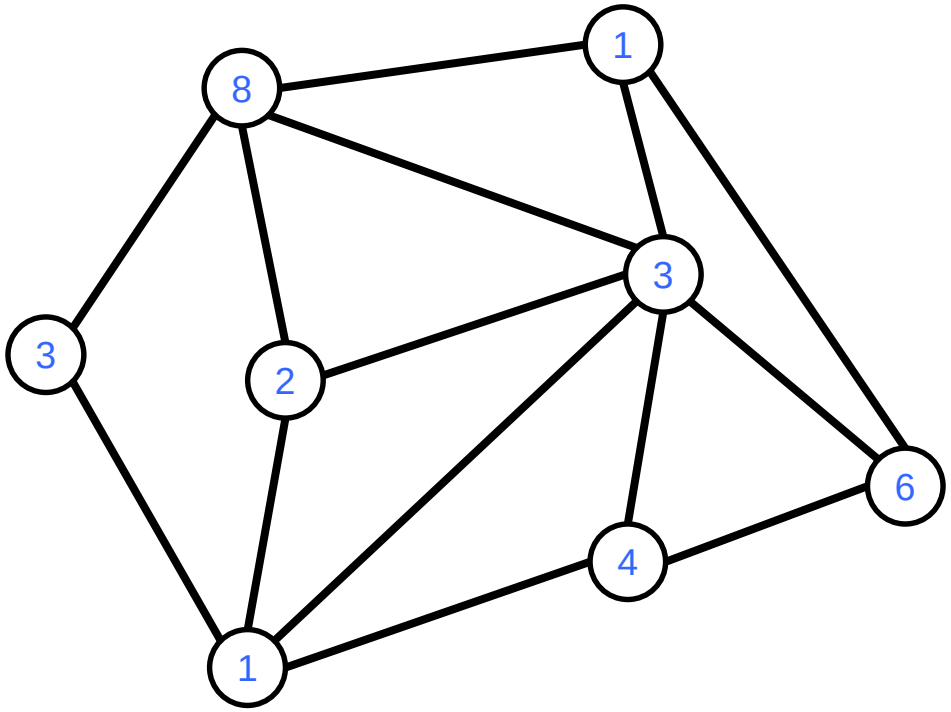
問題：最大サイズの独立頂点集合を求めよ。



重み付き最大独立頂点集合問題

入力：グラフ $G=(V, E)$ （各頂点に重みが付いている）

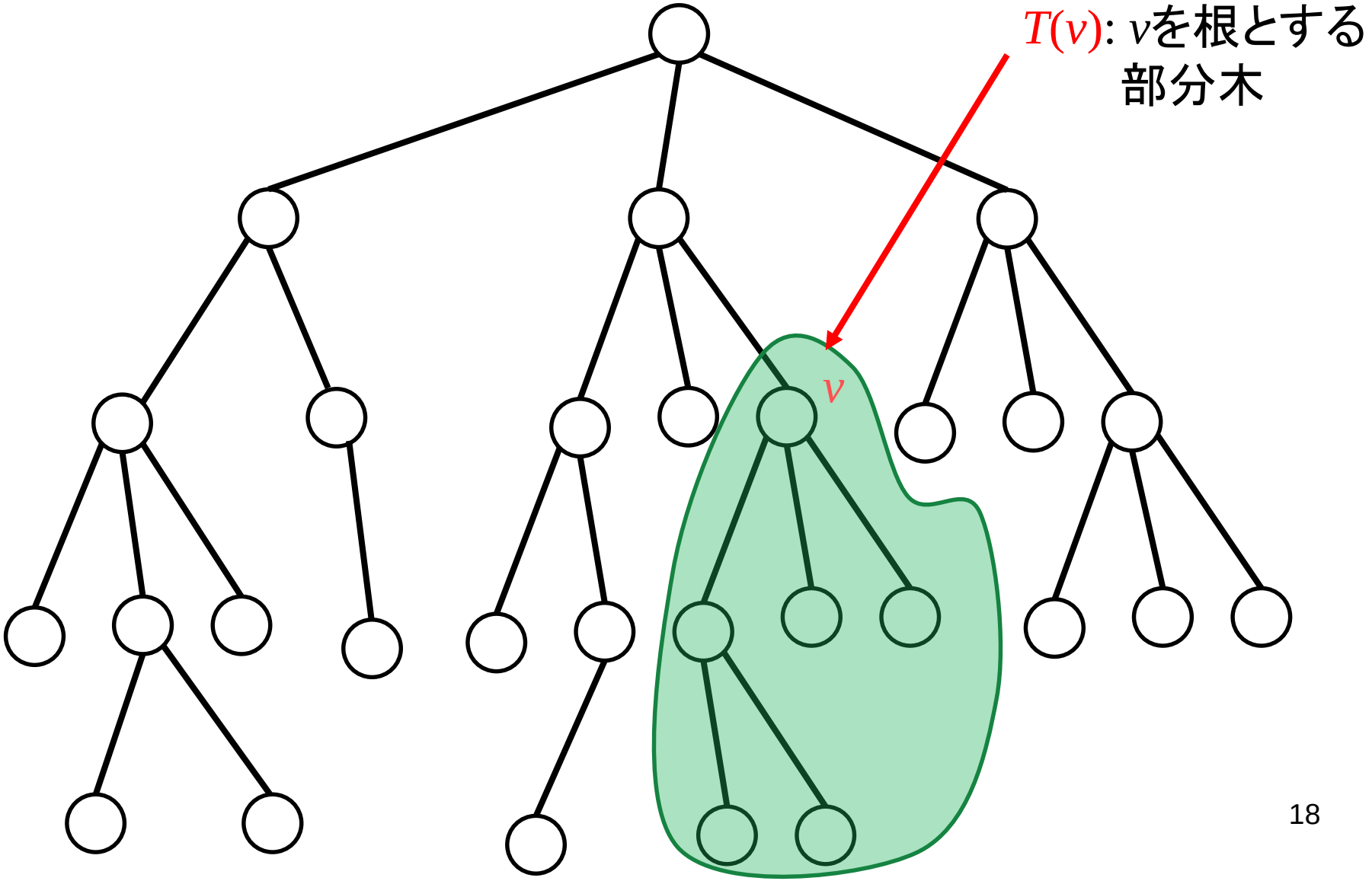
問題：最大サイズの独立頂点集合を求めよ。
（ただしここでは、「サイズ」とは頂点の重みの和。）



重み無しの場合は、木では貪欲アルゴリズムで多項式時間で解けることを見た。
重み付きの場合は、貪欲アルゴリズムはうまく働かない。

問題：貪欲アルゴリズムがうまく働かない木の例を挙げよ。

動的計画法を使うと、重み付きでも（木に限定すれば）
多項式時間で解ける。

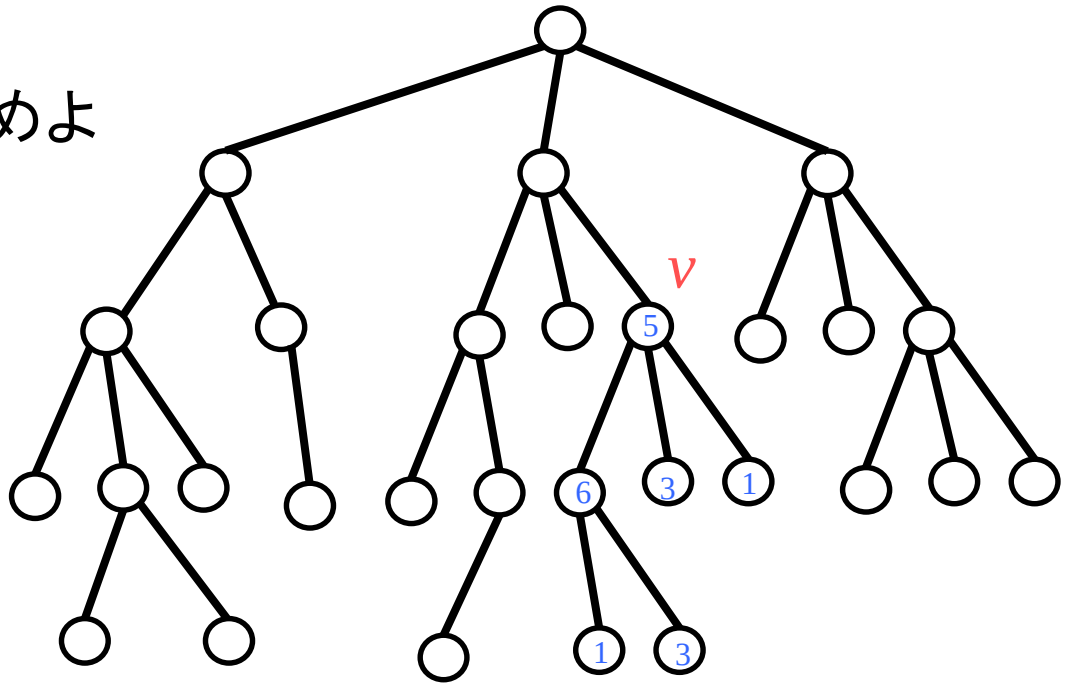


$A(v)$:「vを選ぶ」という条件の下での、
$T(v)$の最大独立頂点集合のサイズ。

$B(v)$:「vを選ばない」という条件の下での、
$T(v)$の最大独立頂点集合のサイズ。

$C(v)$:（何の条件もなく）、$T(v)$の最大独立頂点集合のサイズ。

問題:今の場合の
$A(v)$、$B(v)$、$C(v)$を求めよ



$A(v)=9$

$B(v)=10$

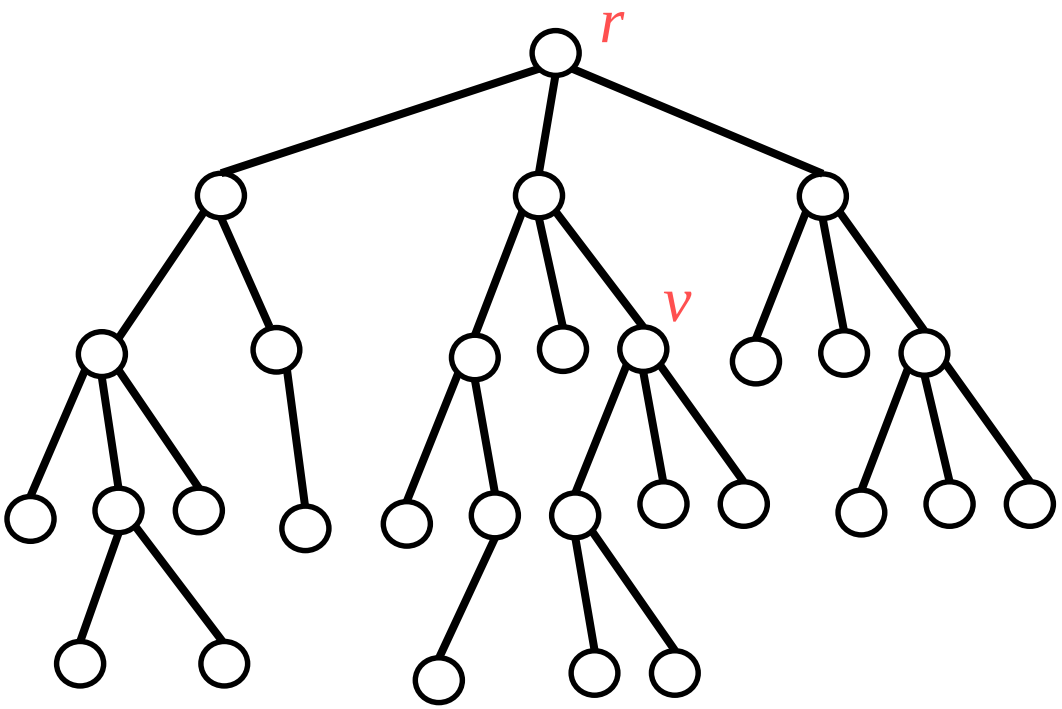
$C(v)=\max\{A(v), B(v)\}=10$

$A(v)$: 「vを選ぶ」という条件の下での、
$T(v)$の最大独立頂点集合のサイズ。

$B(v)$: 「vを選ばない」という条件の下での、
$T(v)$の最大独立頂点集合のサイズ。

$C(v)$: （何の条件もなく）、$T(v)$の最大独立頂点集合のサイズ。

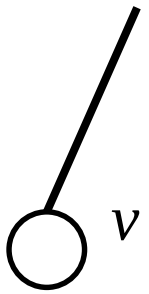
根をrとすると、
元の問題の最終的な
答えは$C(r)$



動的計画法の方針：葉から根に向かって、$A(v)$、$B(v)$、$C(v)$を計算していく。

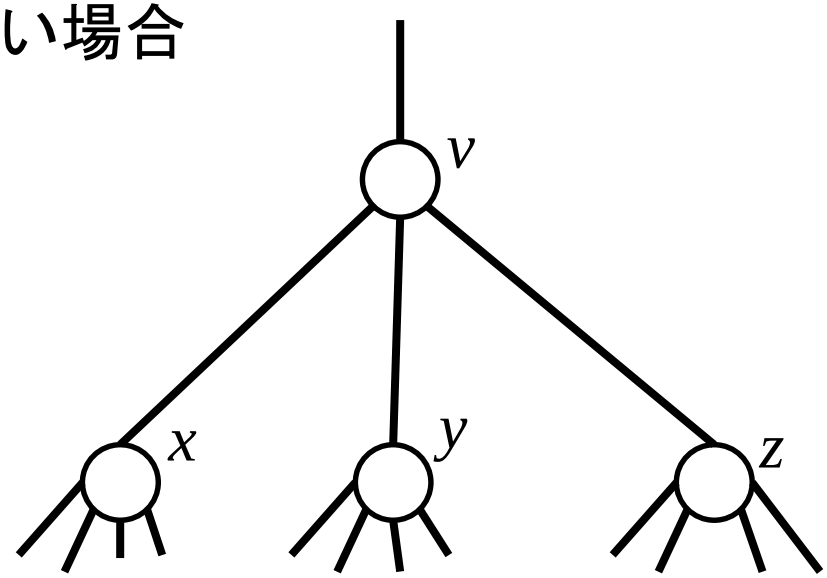
以降、頂点vの重みを$w(v)$と書くことにする。

vが葉の場合



$T(v)$はv1個からなるので、$A(v)= w(v)$, $B(v)=0$,
$C(v)=\max\{A(v), B(v)\}= w(v)$.

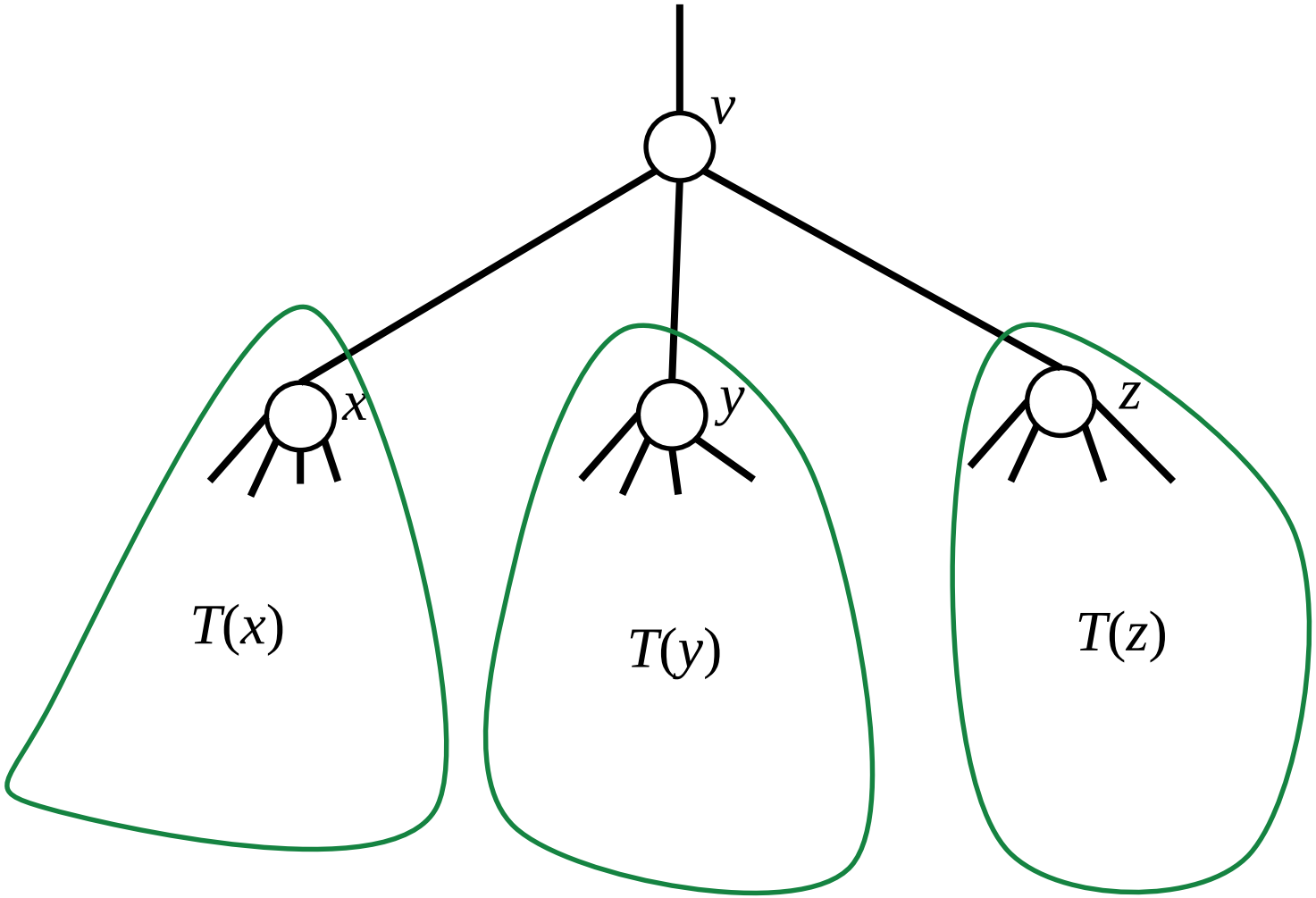
vが葉でない場合



$A(v)=w(v)+B(x)+B(y)+B(z)$
（vを選ぶとすると、x, y, zは選べない。）

$B(v)=C(x)+C(y)+C(z)$
（vを選ばないとすると、x, y, zは選んでも選ばなくても良い。）

$C(v)=\max\{A(v), B(v)\}$



$T(x)$, $T(y)$, $T(z)$ 間にはお互いに枝がなく、またvとも枝がないので、それぞれ独立に計算できる。

$d(v)$: 頂点vの次数

計算時間

$A(v) = w(v) + B(x) + B(y) + B(z)$　$d(v)$-1個の表参照と$d(v)$回の足し算

$B(v) = C(x) + C(y) + C(z)$　$d(v)$-1個の表参照と$d(v)$-1回の足し算

$C(v) = \max\{A(v), B(v)\}$　1回の大小比較

頂点vに関する表計算に$O(d(v))$時間

全体では、$\Sigma d(v) = 2|E| = 2(n-1)$　（木は $|E| = n-1$）

よって、計算時間は$O(n)$

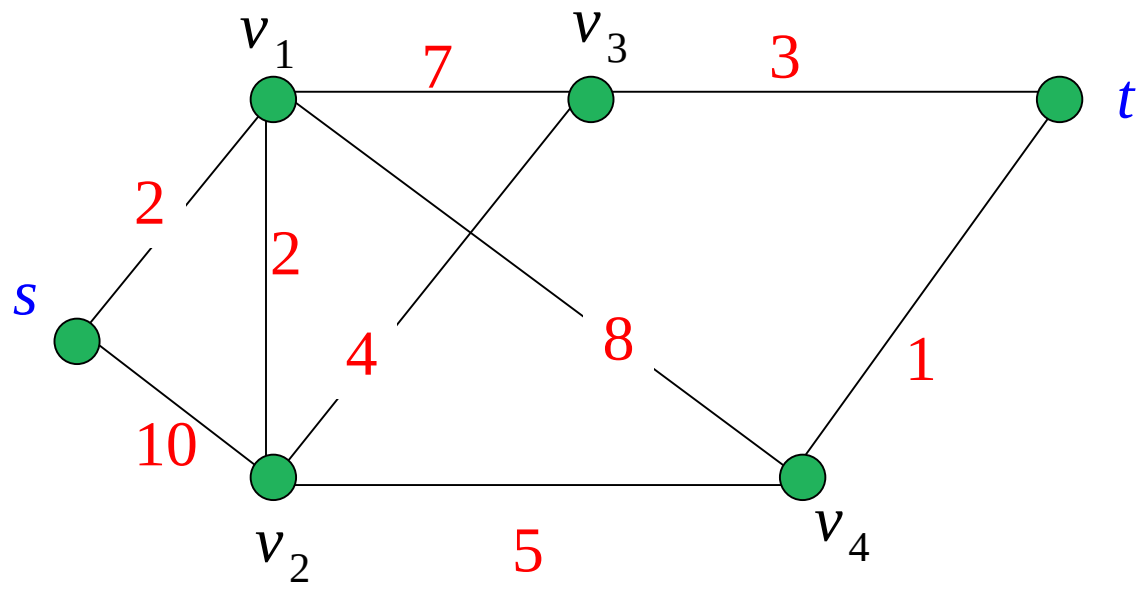
最短経路問題

入力：重みつきグラフ $G=(V, E)$
Vの2頂点 sとt

出力：sからtへの最短経路

問題：sからtへの最短経路は？
その距離は？

例：



単純な方法

- 全てのs-tパスを列挙。
- それぞれの経路のコストを計算。
- 最も短い経路を解として出力。

s —— v_1 —— v_3 —— t 12

s —— v_2 —— v_4 —— t 16

s —— v_1 —— v_4 —— t 11

⋮

s —— v_1 —— v_2 —— v_4 —— t 10

⋮

原理的には最適解が求まる。しかし。。。

- 全てのパスを簡単に列挙できる？
- 出来たとしても、そもそもパスが指数個あったら、多項式時間アルゴリズムにはならない。

問題: sからtへのパスの数が(頂点数の)指数個ある例はあるか？

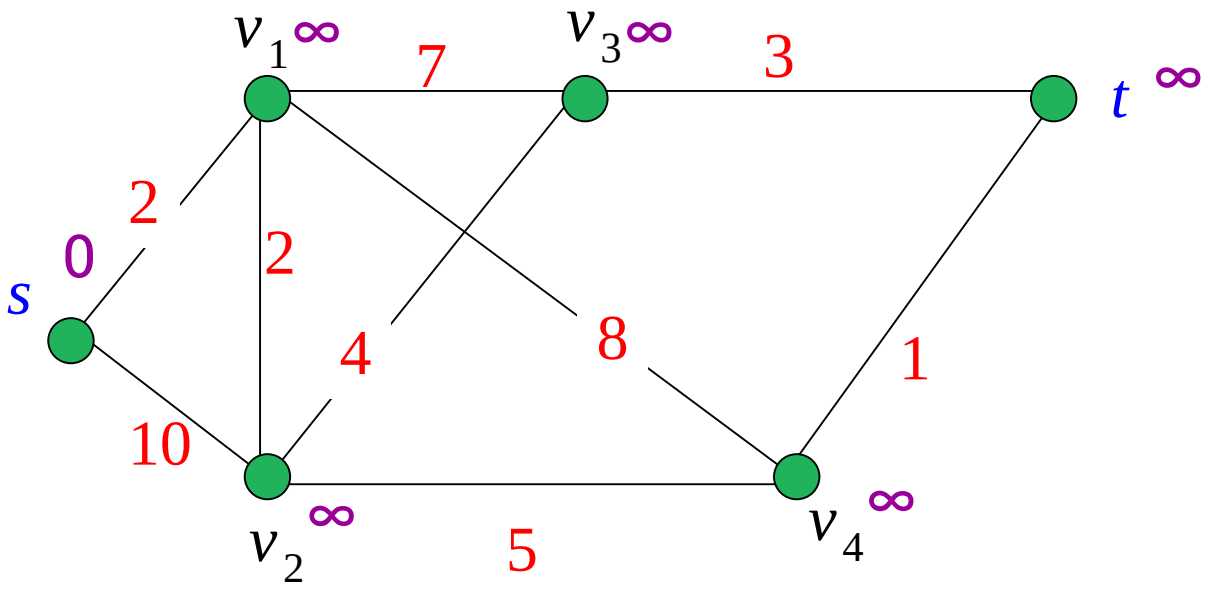
Dijkstra（ダイクストラ）のアルゴリズム

ステップ1：$L=\Phi$（空集合）とする。

各頂点（v）に値（$\delta(v)$）をつける。

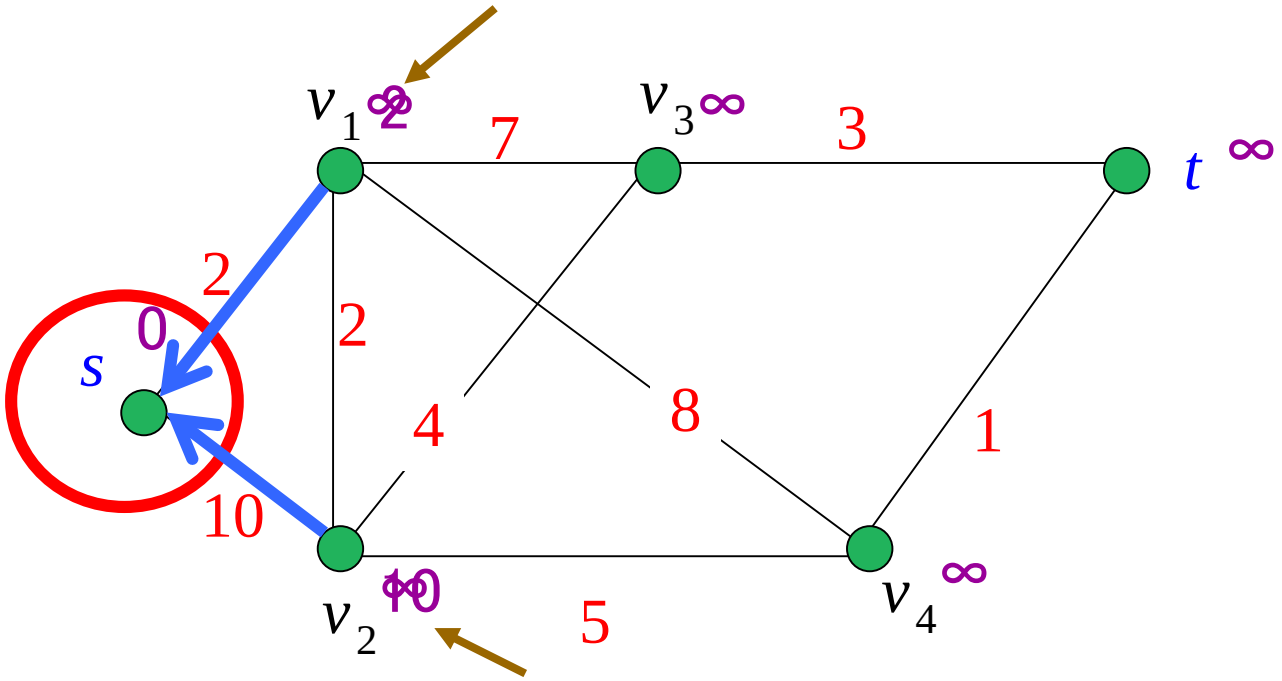
$\delta(s)=0$

$\delta(v)=\infty$（s以外の頂点v）



Dijkstra（ダイクストラ）のアルゴリズム

ステップ2：Lに頂点sを加える。
sから到達可能な頂点の値を$d(s,v)$に更新
（更新された頂点は、sにリンクを張る）



Dijkstra（ダイクストラ）のアルゴリズム

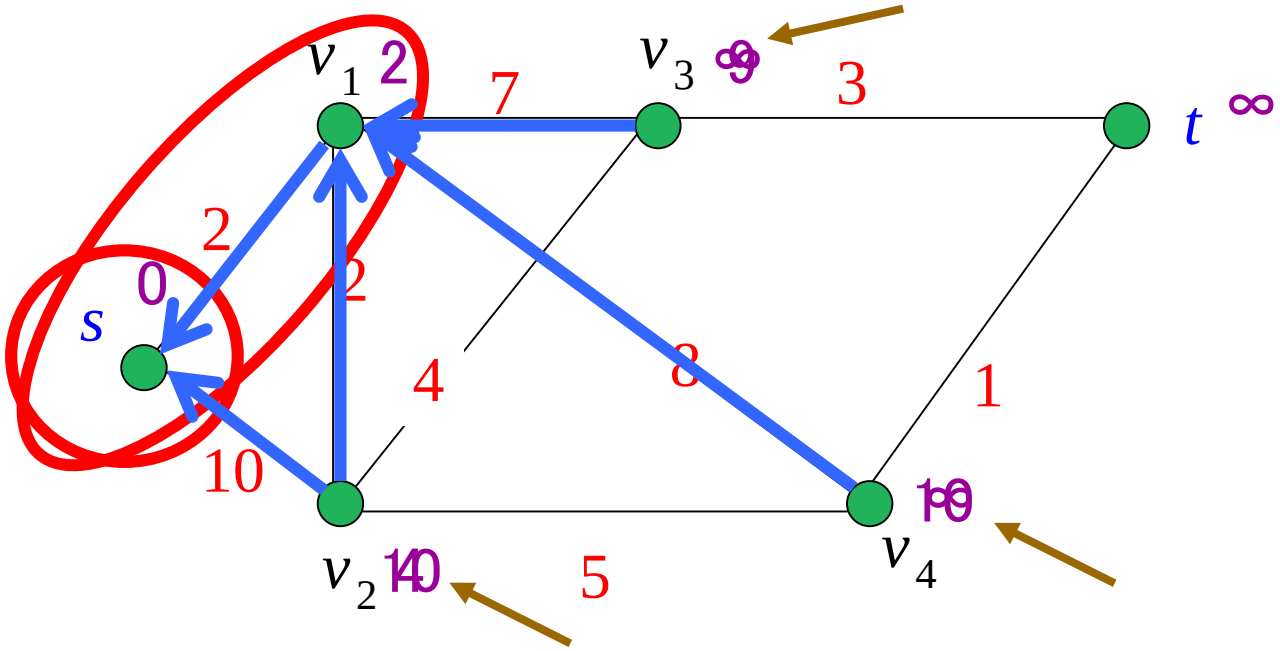
ステップ3：tがLに入るまで、以下を繰り返す。

- Lに入っていない頂点の中で、値が最小のものをvとする。
- vをLに加える。
- Lに入っていない、vに隣接する全ての頂点uに対して、

$$\delta(u) = \min\{\delta(u), \delta(v) + d(v,u)\}$$

とする。

- 値が更新されたものはリンクを更新する。



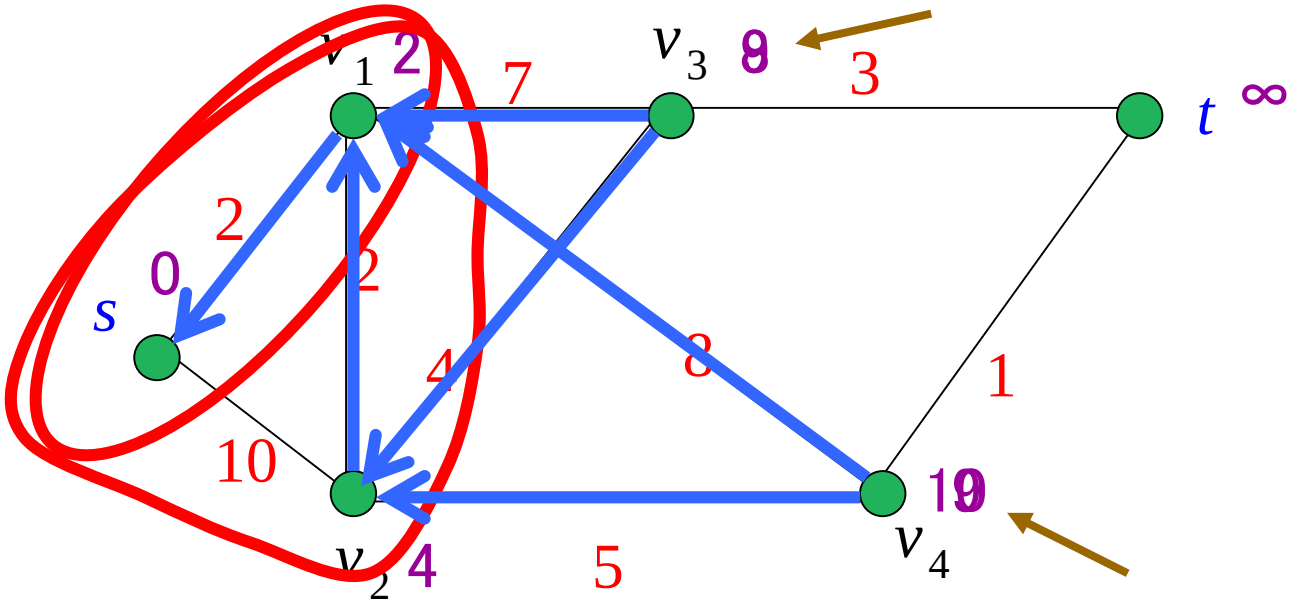
Dijkstra（ダイクストラ）のアルゴリズム

ステップ3：tがLに入るまで、以下を繰り返す。

- Lに入っていない頂点の中で、値が最小のものをvとする。
- vをLに加える。
- Lに入っていない、vに隣接する全ての頂点uに対して、

$$\delta(u) = \min\{\delta(u), \delta(v)+d(v,u)\}$$とする。

- 値が更新されたものはリンクを更新する。



Dijkstra（ダイクストラ）のアルゴリズム

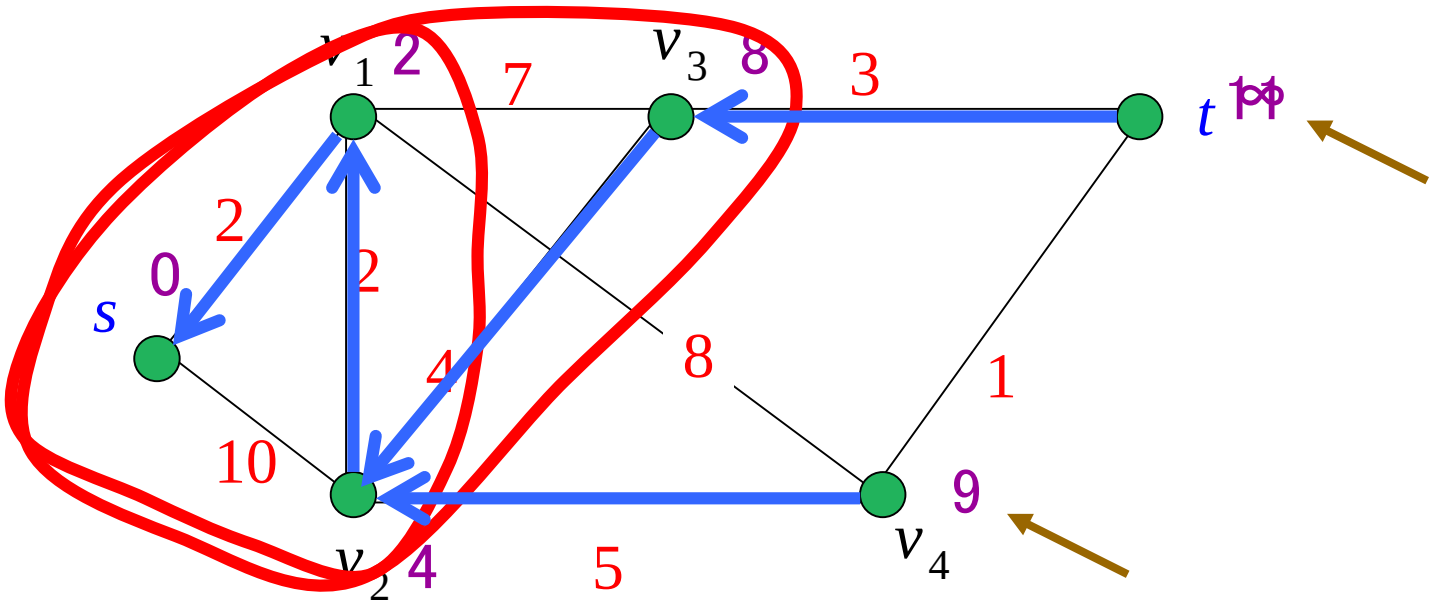
ステップ3：tがLに入るまで、以下を繰り返す。

- Lに入っていない頂点の中で、値が最小のものをvとする。
- vをLに加える。
- Lに入っていない、vに隣接する全ての頂点uに対して、

$$\delta(u) = \min\{\delta(u), \delta(v)+d(v,u)\}$$

とする。

- 値が更新されたものはリンクを更新する。



Dijkstra（ダイクストラ）のアルゴリズム

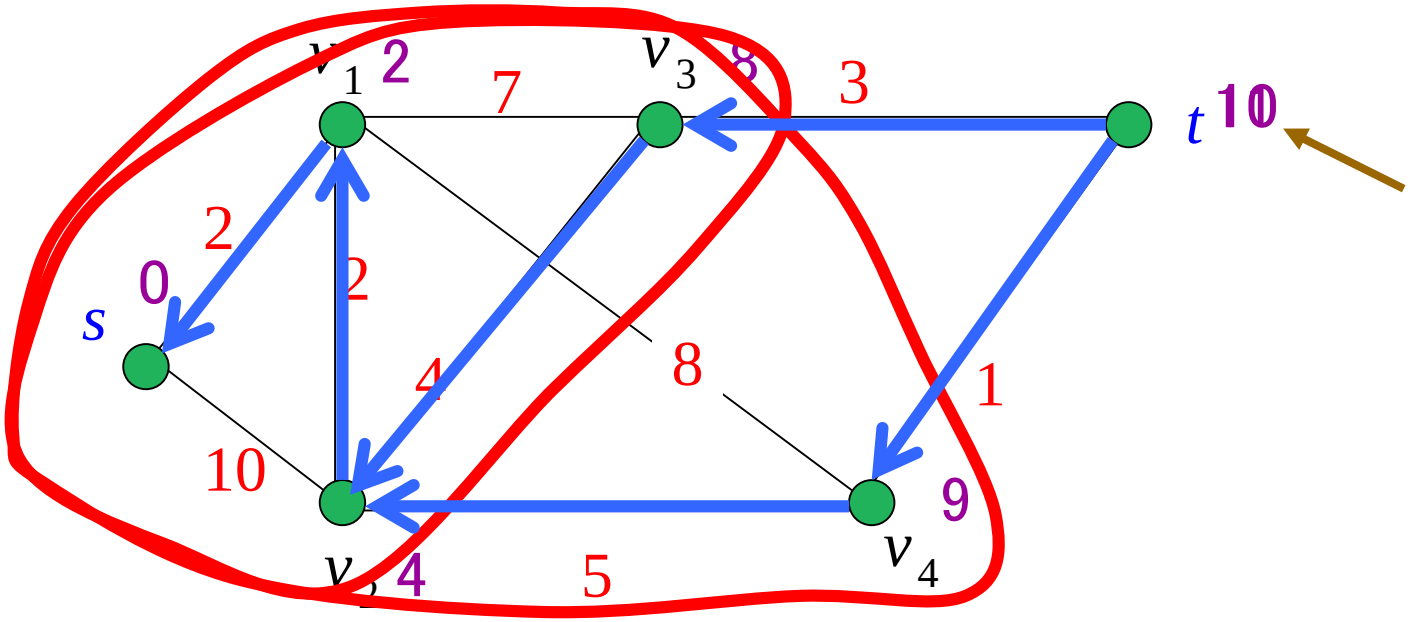
ステップ3：tがLに入るまで、以下を繰り返す。

- Lに入っていない頂点の中で、値が最小のものをvとする。
- vをLに加える。
- Lに入っていない、vに隣接する全ての頂点uに対して、

$$\delta(u) = \min\{\delta(u), \delta(v)+d(v,u)\}$$

とする。

- 値が更新されたものはリンクを更新する。



Dijkstra（ダイクストラ）のアルゴリズム

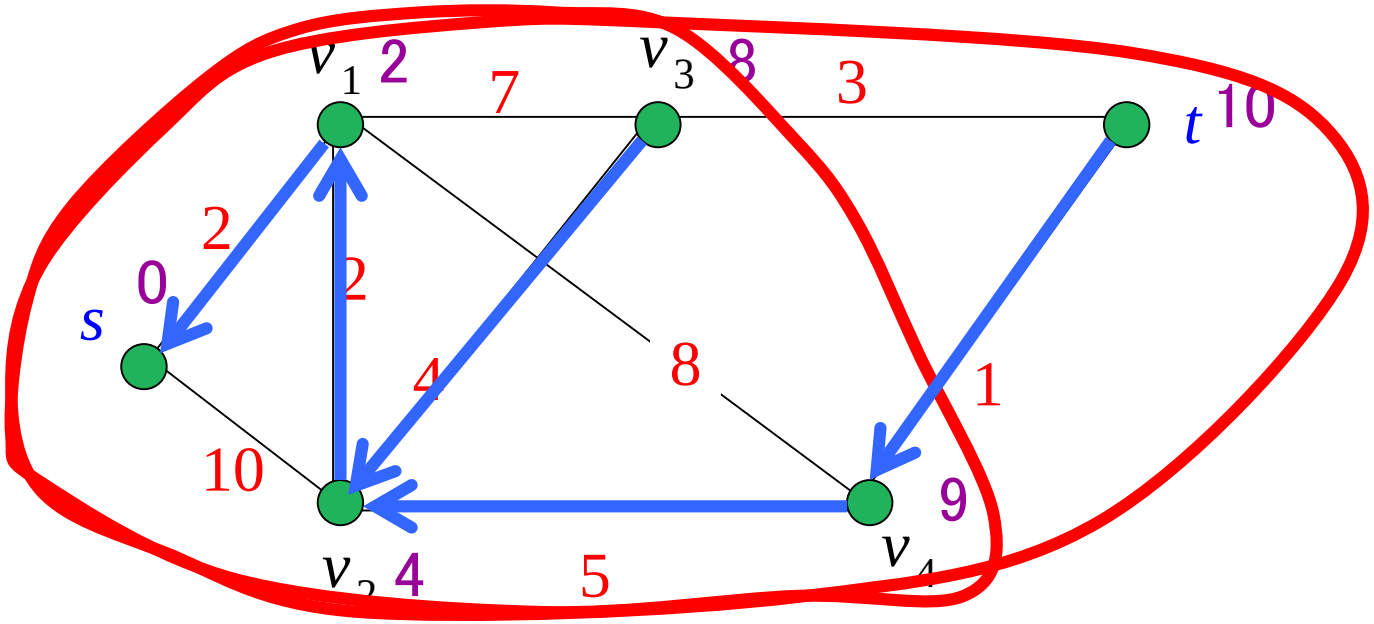
ステップ3：tがLに入るまで、以下を繰り返す。

- Lに入っていない頂点の中で、値が最小のものをvとする。
- vをLに加える。
- Lに入っていない、vに隣接する全ての頂点uに対して、

$$\delta(u) = \min\{\delta(u), \delta(v)+d(v,u)\}$$

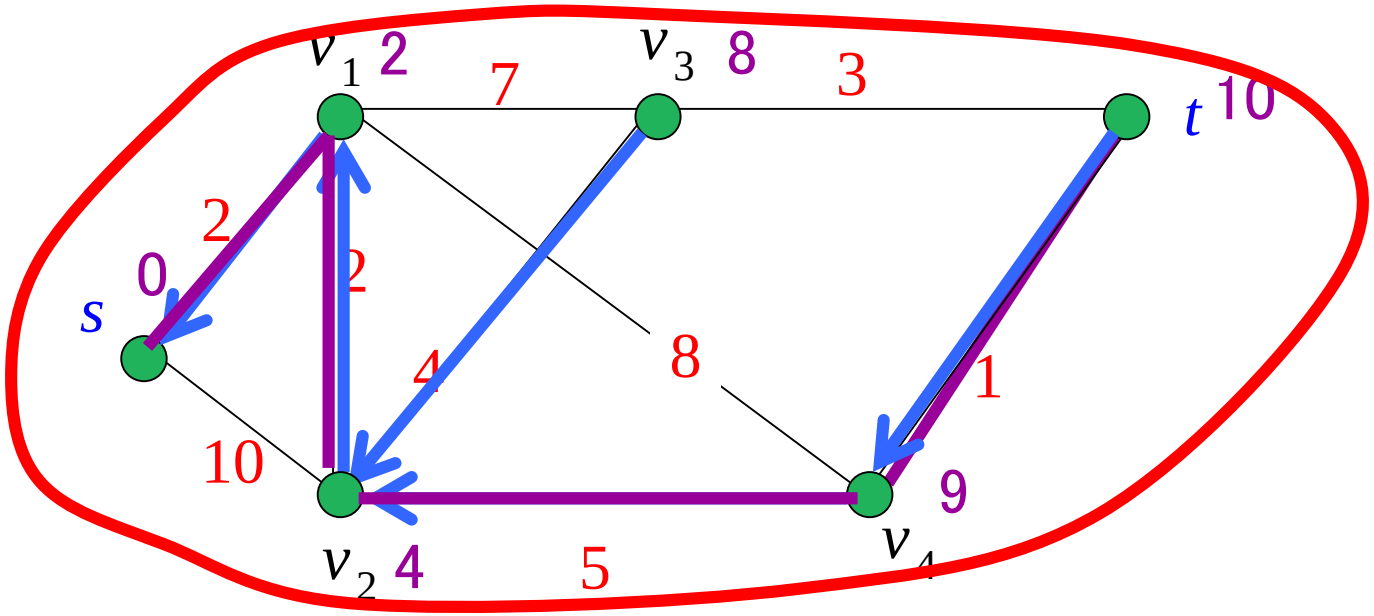
とする。

- 値が更新されたものはリンクを更新する。



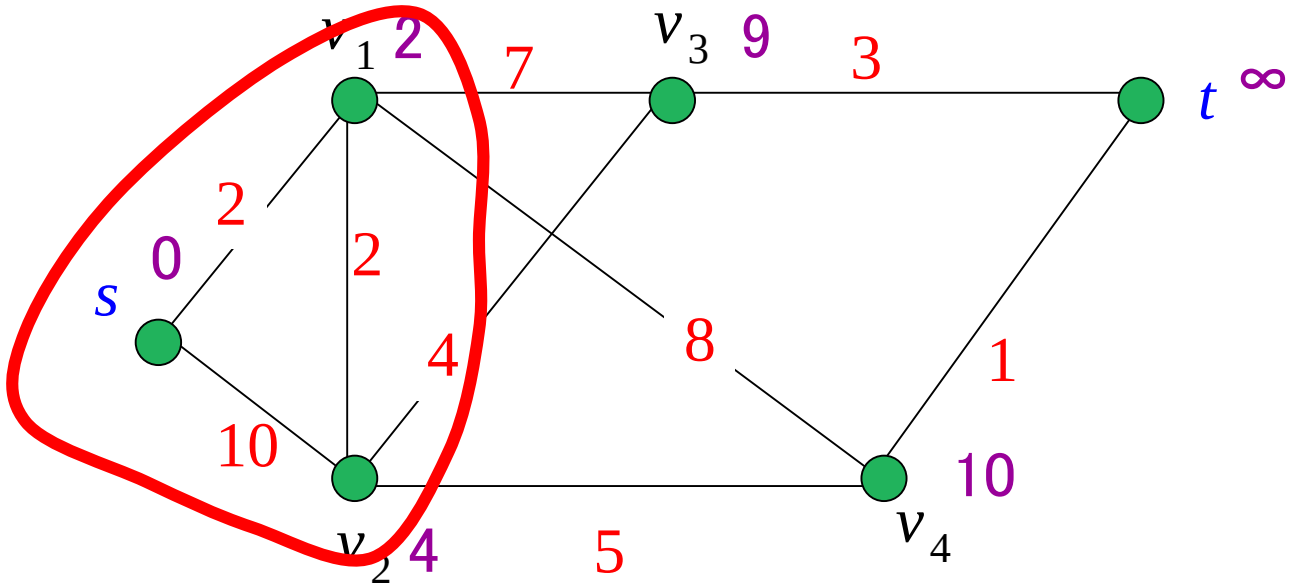
Dijkstra（ダイクストラ）のアルゴリズム

ステップ4：この時点でtについている値が、sからtまでの
最短距離。
tにその値を付けるに至った経路が最短経路。



正しさの証明

アルゴリズムの任意の時点で、
　「Lに入っている頂点の値は、sからそこまでの最短距離」
を証明する。
つまり、



帰納法で証明する

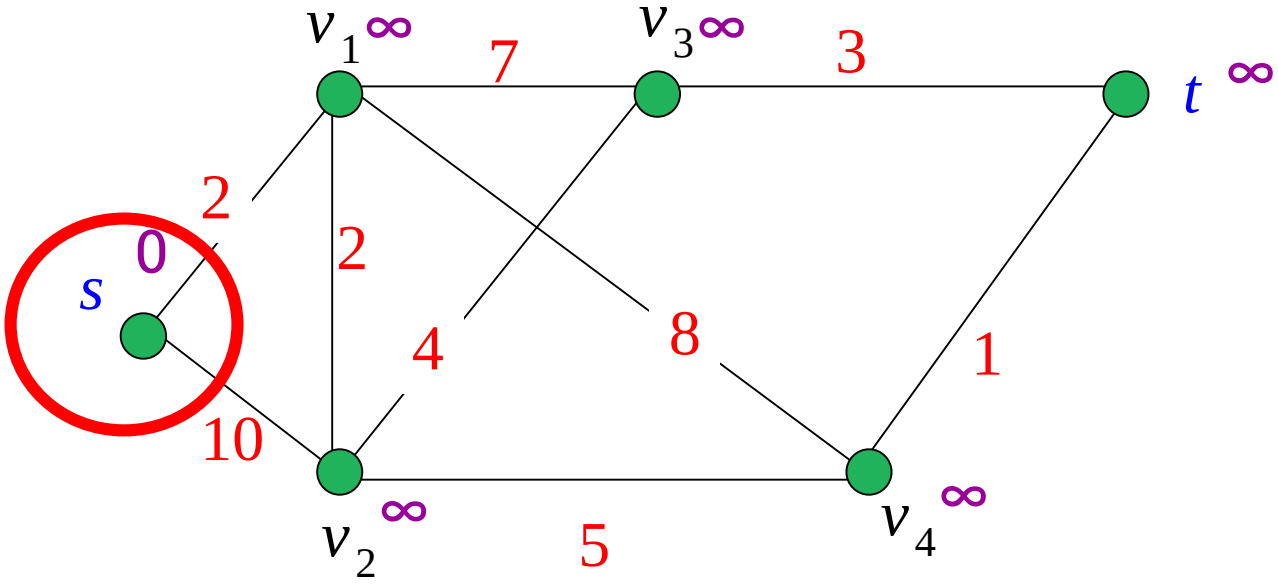
最初:

*L*に入っているのは*s*のみ。

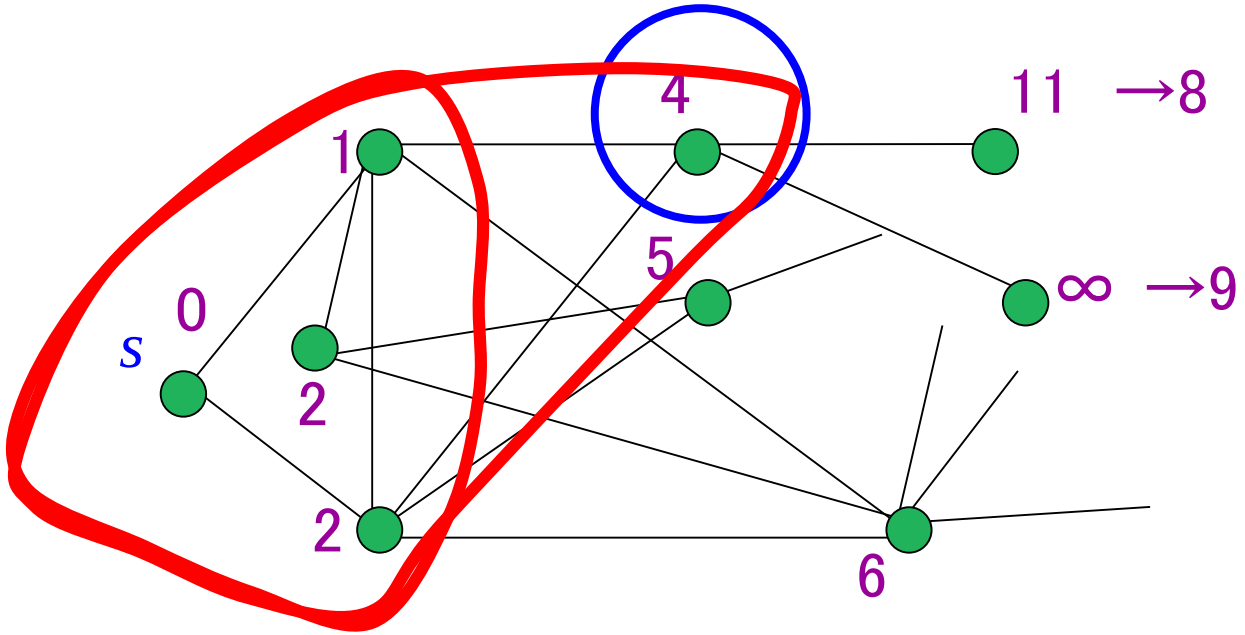
*s*の値は0。

*s*から*s*までの最短距離は0。

よって成り立つ。

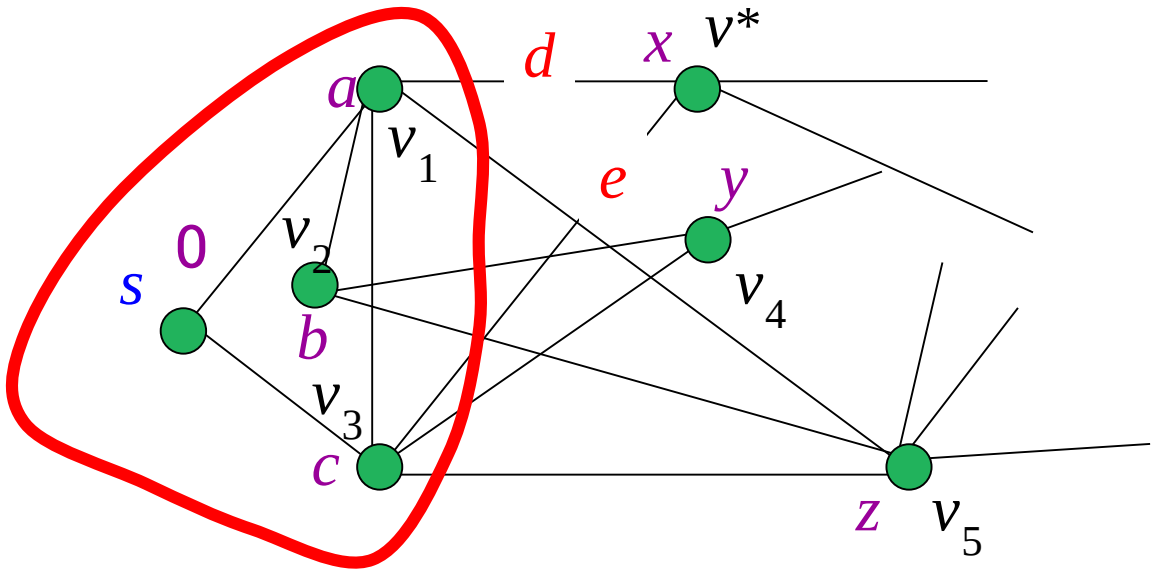


あるステップで正しいとする



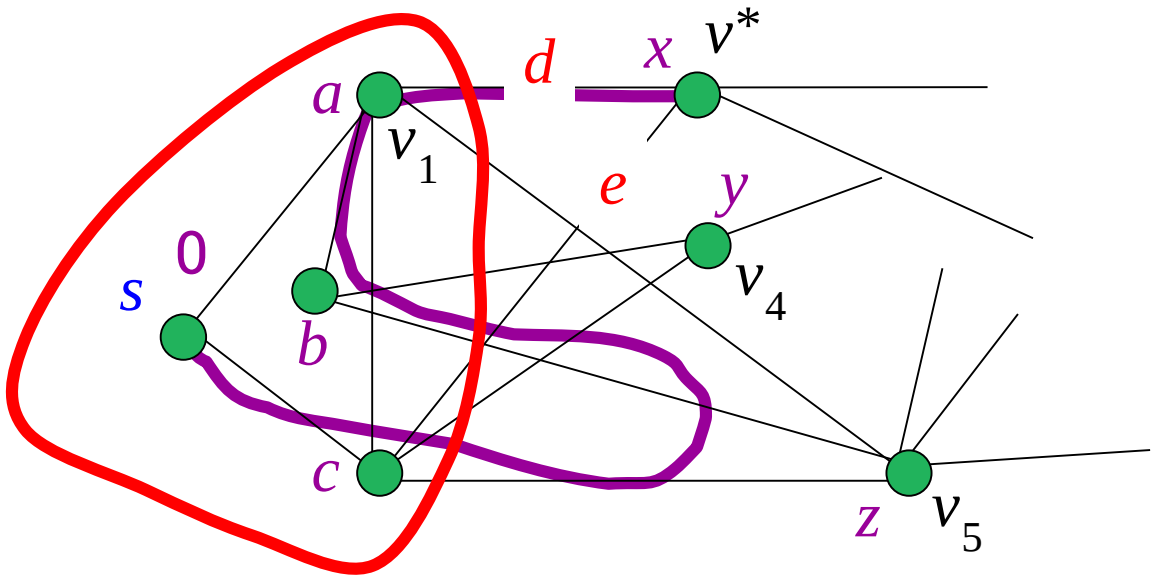
このとき、新たに加えられた頂点についても正しいことを示す。

いま$x \leqq y$, $x \leqq z$が成り立っている。(v^*が選ばれたのだから)
$x=\min\{a+d, c+e\}$である。(xの値の決め方から)



sからv^*への最短距離がxでないとして、矛盾を導く。
xより短い $s-v^*$ パスがあったとする。その長さを$x' < x$とする。

場合(1):最後に使われた枝が、Lの中の頂点からv^*に向かう。



x'=(sからv_1までの経路)+d

$x \leqq a+d$

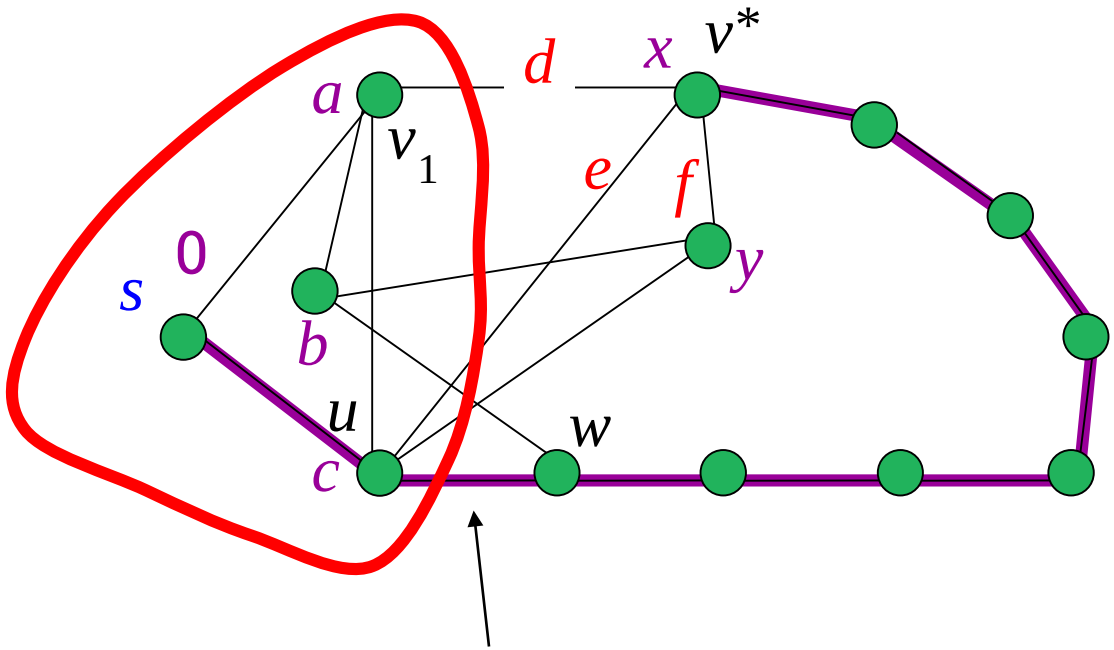
$x' < x$

なので、(sからv_1までの経路)< aとなる。

ということは、sから(Lの中にある)v_1への最短距離が

aであること(つまり帰納法の仮定)に矛盾。

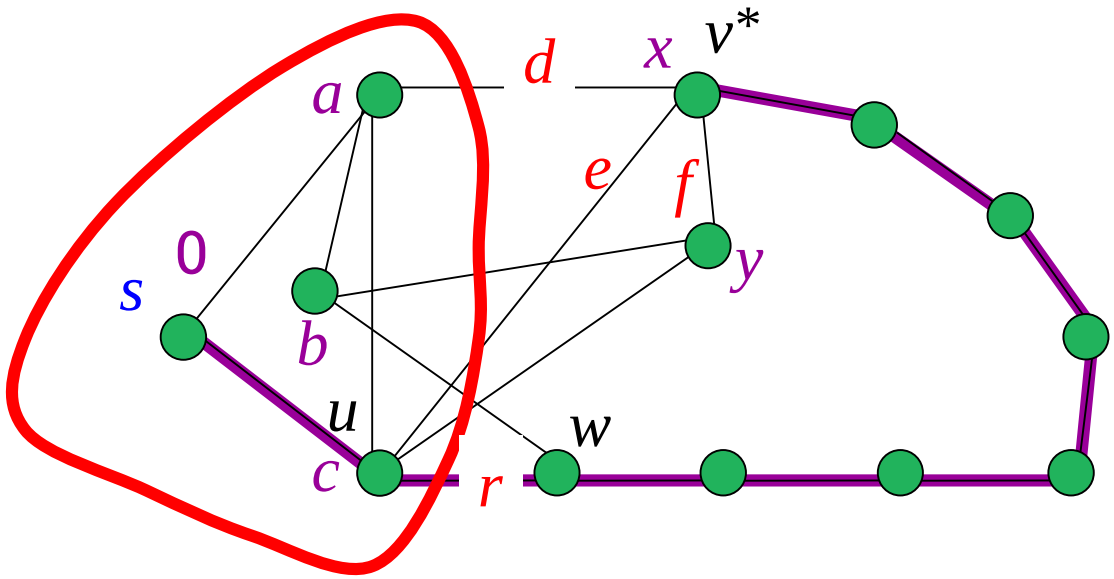
場合(2):最後に使われた枝が、Lの外の頂点からv^*に向かう。



v^*から逆にたどって、初めてLに入るところ。

sからuへの最短経路は、帰納法の仮定よりc。
なので、紫の道も、その最短路を通っているはず。
(さもなければ、sからv^*へ、紫よりも短い道がある。)

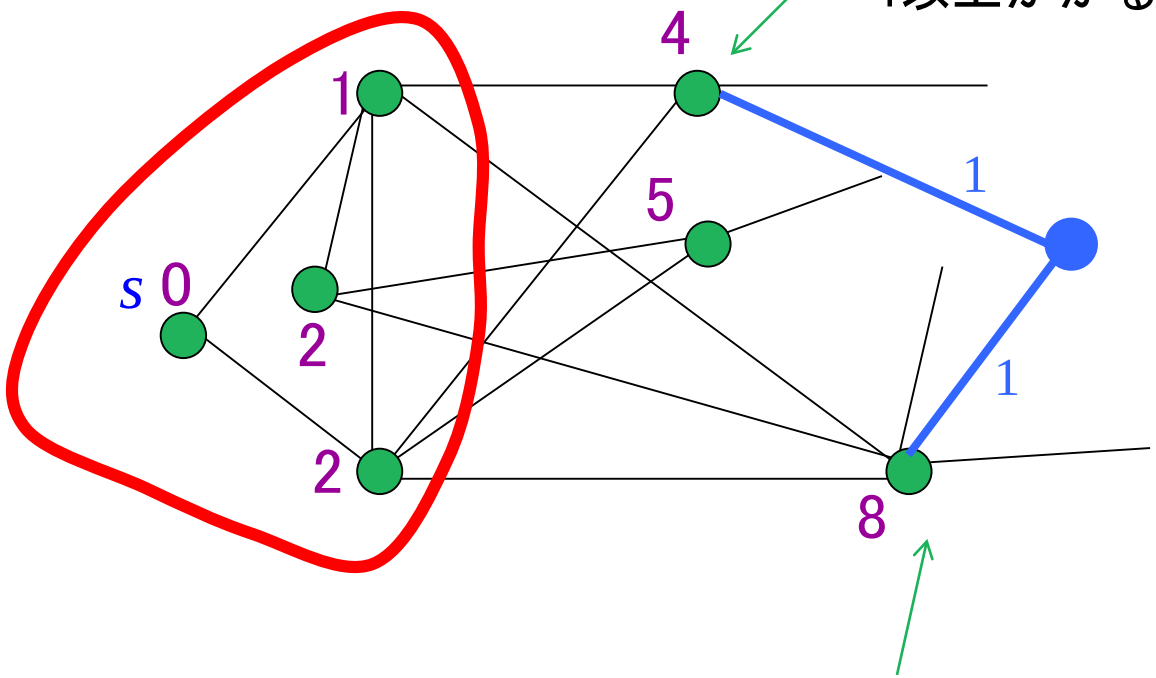
場合(2):最後に使われた枝が、Lの外の頂点からv^*に向かう。



$s \to u \to w$路は、x'より短い。(紫がx'なのだから) つまり$c+r < x'$。
wはuにつながっているので、uがLに入った時点で$\delta(w)$は
$c+r$以下であるはず。
$\delta(w) \leqq c+r < x' < x$ つまり $\delta(w)<x$。
したがって、次のステップでv^*がLに入れられるのはおかしい。
矛盾。

直観的には

これは確定
4より短くなりようがない
（他の頂点へ、赤の中から行くのは、
4以上かかるから）



こいつは確定してはいけない
8より短い道があるかも

計算時間

頂点はn個あり、それぞれが高々n回しか更新されない。

$O(n^2)$

枝はm本あり、それぞれ1回しか走査されない。

$O(m)$

よって全体で$O(n^2+m)$。

問題：次のグラフの最短経路とその値を求めよ