アルゴリズムと データ構造

コンピュータサイエンスコース
知能コンピューティングコース

第11回
最短路問題のアルゴリズム

塩浦昭義
情報科学研究科 准教授

shioura@dais.is.tohoku.ac.jp
http://www.dais.is.tohoku.ac.jp/~shioura/teaching

最も短い経路を求める

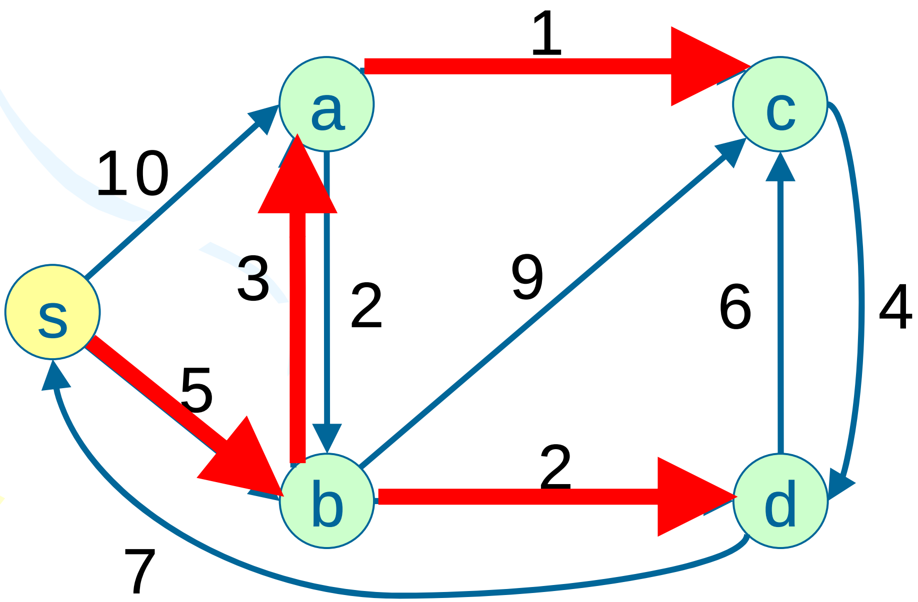
仙台駅から
勾当台公園までの
最短経路を
求めたい
➔グラフを使って
　モデル化

各頂点の間に距離
(移動時間)を与える

最短路問題

- 入力：有向グラフ G=(V, E)
  各枝の長さ d(e) (e∈E), 始点 s∈V
- 出力：s から V へのすべての頂点 v への最短路

（sからvへの最短路 = sからvへの路（パス）のうち，
路に含まれる枝の長さの和が最小のもの）



この授業では，
枝の長さは
すべて非負と仮定

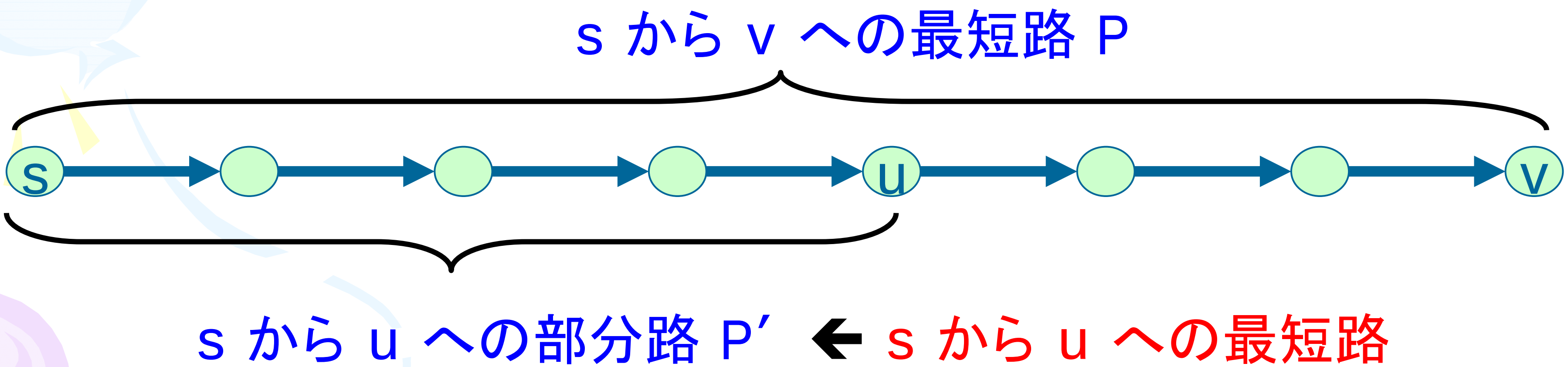
最短路の部分路は最短路

性質1: P は s から v への最短路とする.

P の途中に頂点 u が含まれると仮定する.

このとき, s から u への P の部分路 P' は,

s から u への最短路である.



最短路の部分路は最短路

性質1： P は s から v への最短路とする.

P の途中に頂点 u が含まれると仮定する.

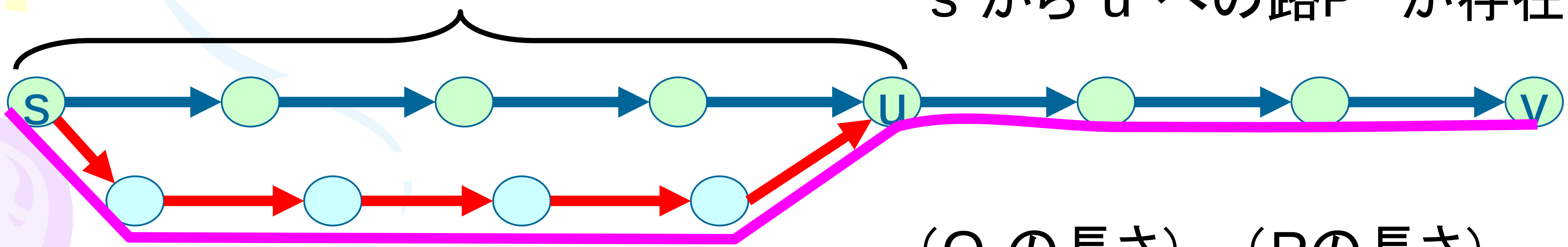
このとき, s から u への P の部分路 P’ は,

s から u への最短路である.

証明:

P’ が s から u への
最短路ではないと仮定
➔ P’ より短い
s から u への路P’’ が存在

s から u への部分路 P′



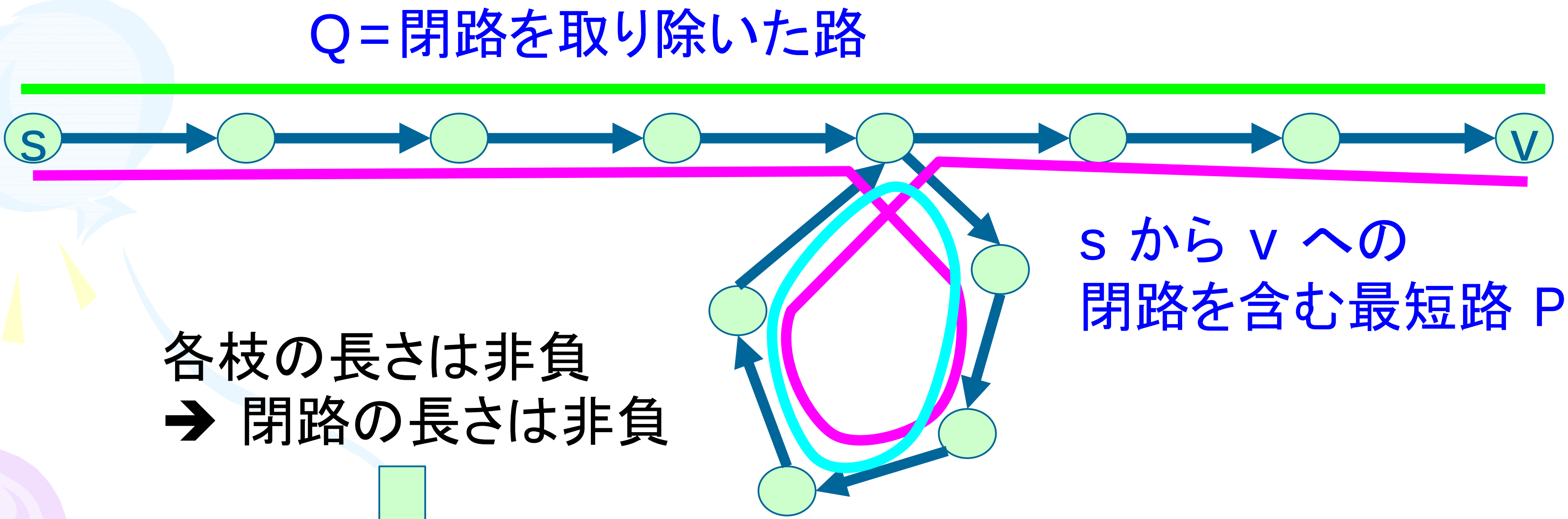
s から u への路 P’’

Q ＝ P’’ を経由して s から v に行く路

(Q の長さ)ー(Pの長さ)
＝(P’’の長さ)ー(P’の長さ)
＜ 0 (矛盾)

閉路を含まない最短路の存在

性質2：ｓ から ｖ への最短路で，閉路を含まないものが存在する



各枝の長さは非負
➔ 閉路の長さは非負

（Ｐの長さ）ー（Ｑの長さ） ＝ 閉路の長さ ≧ ０
➔ ＱはＰより長くない ➔ Ｑも最短路，閉路を含まない

交差しない最短路

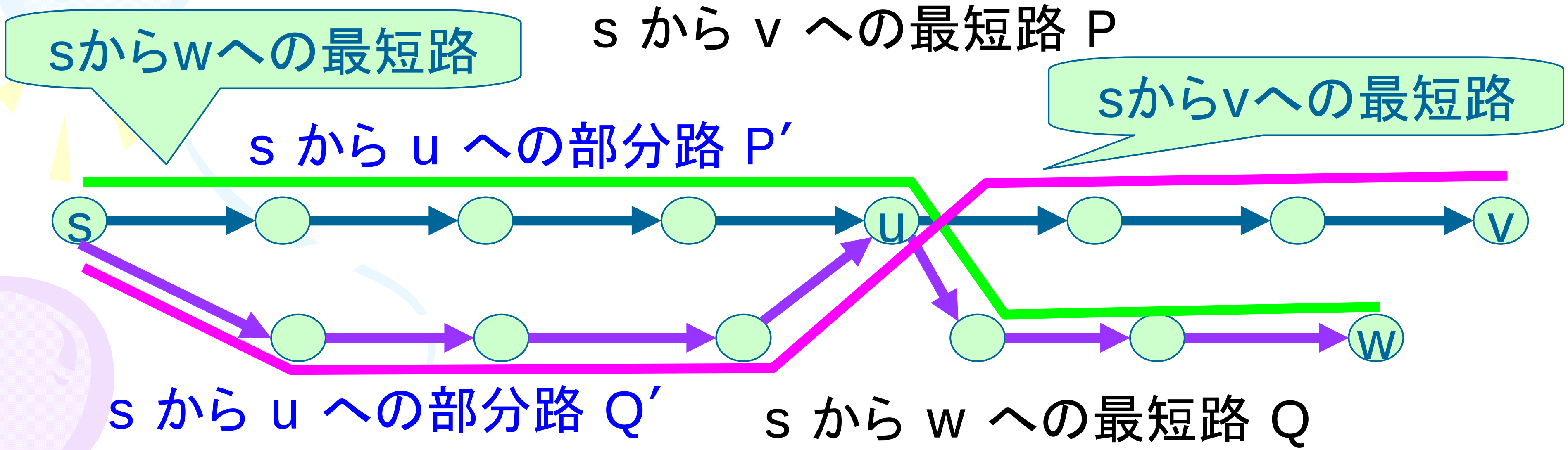
性質3:

P: s から v への最短路, Q: s から w への最短路

P, Q は頂点 u を含むと仮定

P’: s から u への P の部分路, Q’: s から u への Q の部分路

➔ P, Q において P’ と Q’ を入れ替えても, それぞれ最短路



P’ も Q’ も s から u への最短路（性質1） ➔ 2つの長さは等しい

交差しない最短路

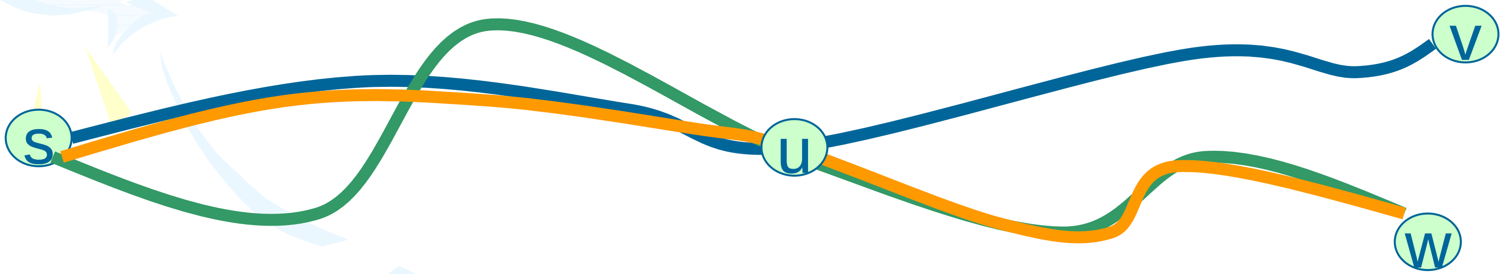
性質3:

P: s から v への最短路, Q: s から w への最短路

P, Q は頂点 u を含むと仮定

P': s から u への P の部分路, Q': s から u への Q の部分路

➔ P, Q において P' と Q' を入れ替えても, それぞれ最短路



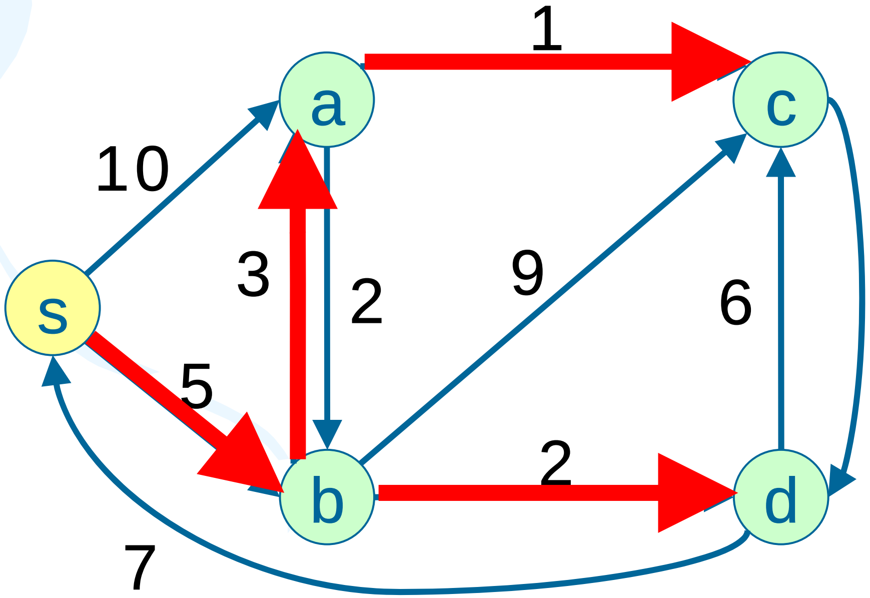
2つの最短路が頂点 u を共有する

➔s から u までの部分路を共有する2つの最短路が存在

最短路木

- 最短路の部分路は最短路
- 閉路を含まない最短路が存在

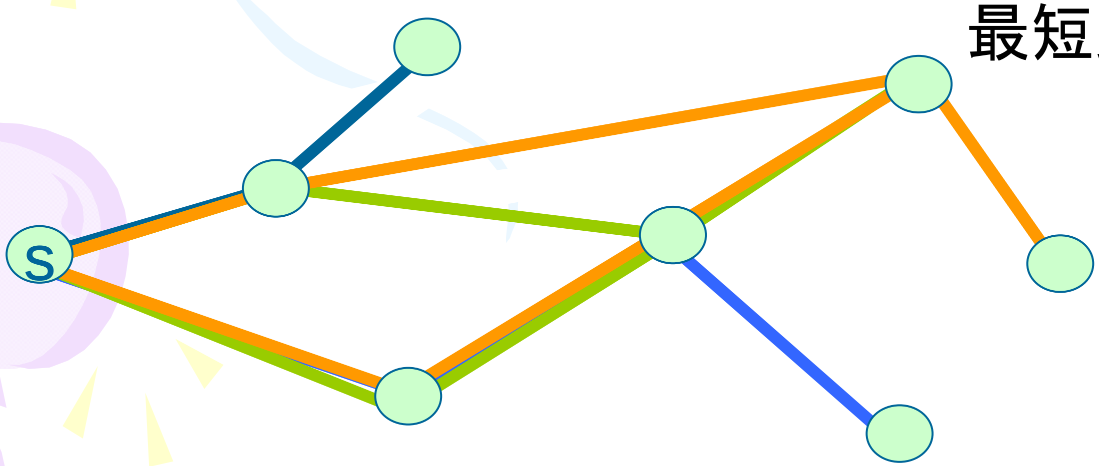
s から全頂点への最短路は木を使って表現できる



最短路木が存在することの証明

- 以下の手順で全頂点への最短路を計算➔最短路木が得られる
  – 最短路が求められていない頂点 v を任意に選ぶ
  – s から v までの(閉路を含まない)最短路 P を求める
    - P が既存の最短路とある頂点 u を共有する
      ➔ s から u までの部分路を共有させるようにして
      最短路を変更する(性質3より)

  ※ P に含まれる全ての頂点に対して,
  最短路が求められたことになる



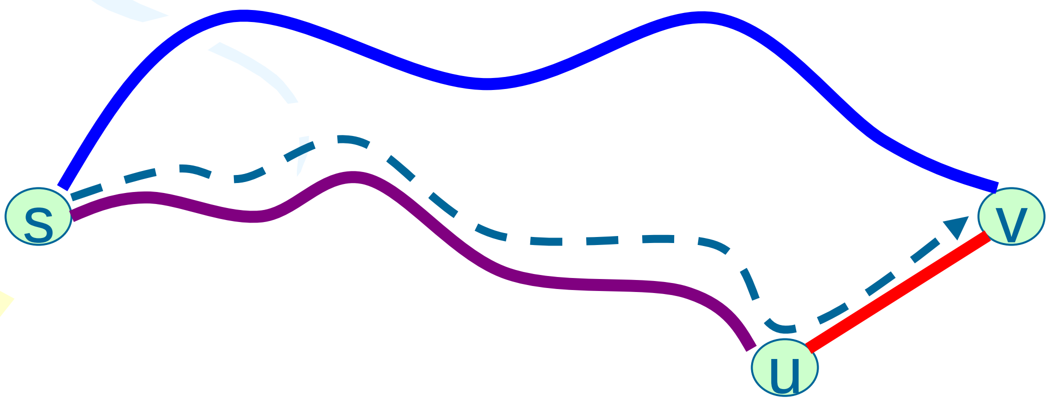
最短路長の性質

- f(v): 頂点 s から頂点 v までの最短路の長さ

➔任意の枝 (u, v) に対して次の不等式が成立

$$f(v) \leqq f(u) + d(u, v)$$

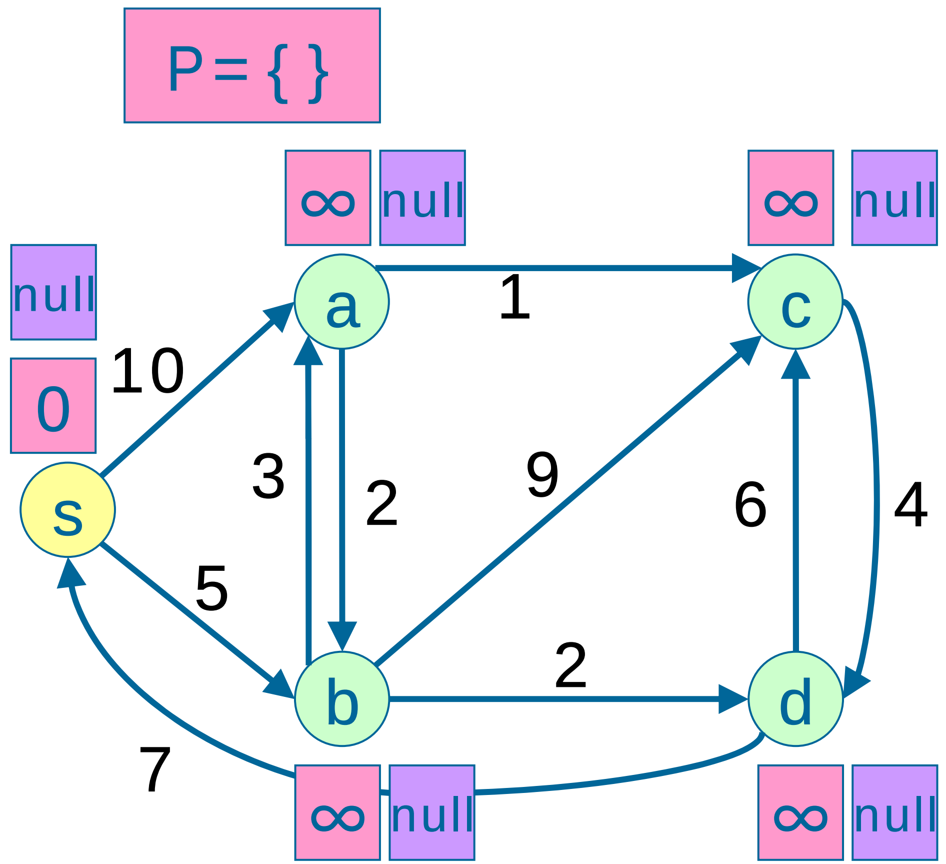
不等式の意味: s から u への最短路を通って, その後 (u,v) を通る路は, s から v への路
➔ その長さ f(u) + d(u, v) は, s から v への最短路の長さ f(v) 以上



ダイクストラのアルゴリズム

- 全ての枝が非負の場合に，頂点 s から全ての頂点への最短路を求めるアルゴリズム

ステップ0:
各頂点への最短路の長さの暫定値をセット
$d^*(s) = 0$
$d^*(v) = +\infty \ (v \in V - \{s\})$
最短路が求められた頂点集合をセット　$P = \{\ \}$
最短路木を表すための配列をセット
$\pi(v) = \text{null} \ (v \in V)$

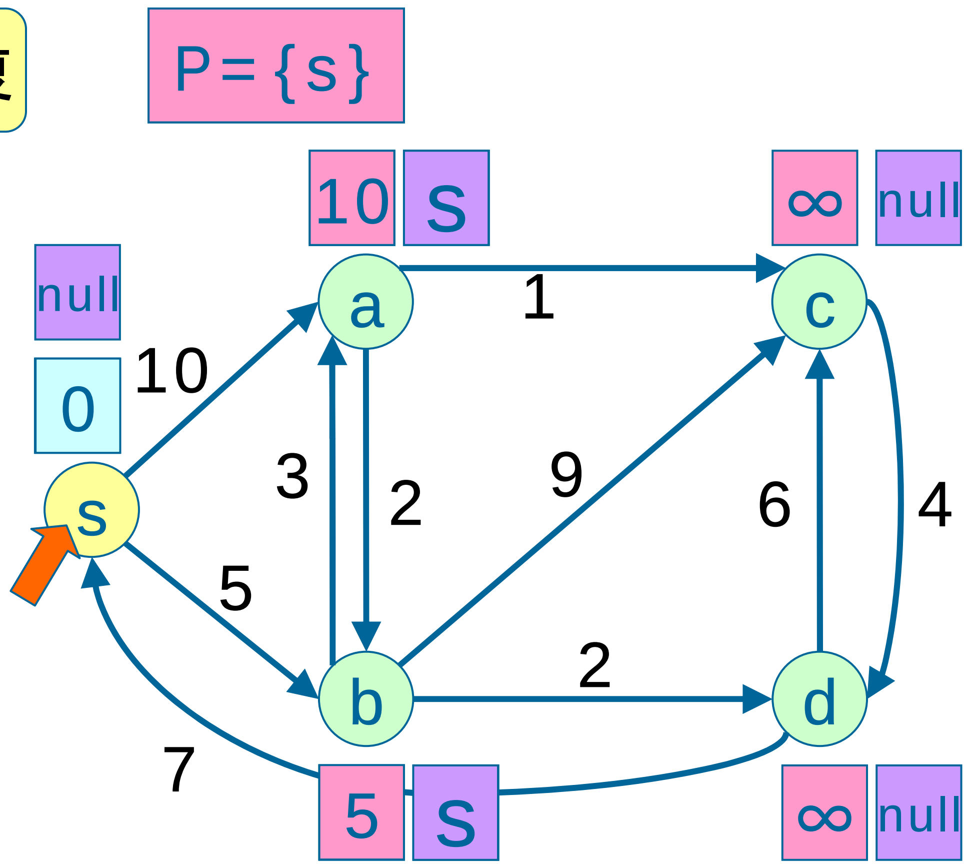


ダイクストラのアルゴリズム

- 全ての枝が非負の場合に，頂点 s から全ての頂点への最短路を求めるアルゴリズム

P=V となるまで反復

ステップ1：
d*(v)が最小な v∈V-Pを選ぶ
➔ u とおく
集合 P の更新：P := P∪{u}
d*(v), π(v) (v∈V-P)の更新
　d*(v) > d*(u)+d(u,v)
　ならば
　　d*(v) = d*(u)+d(u,v)
　　π(v) = u

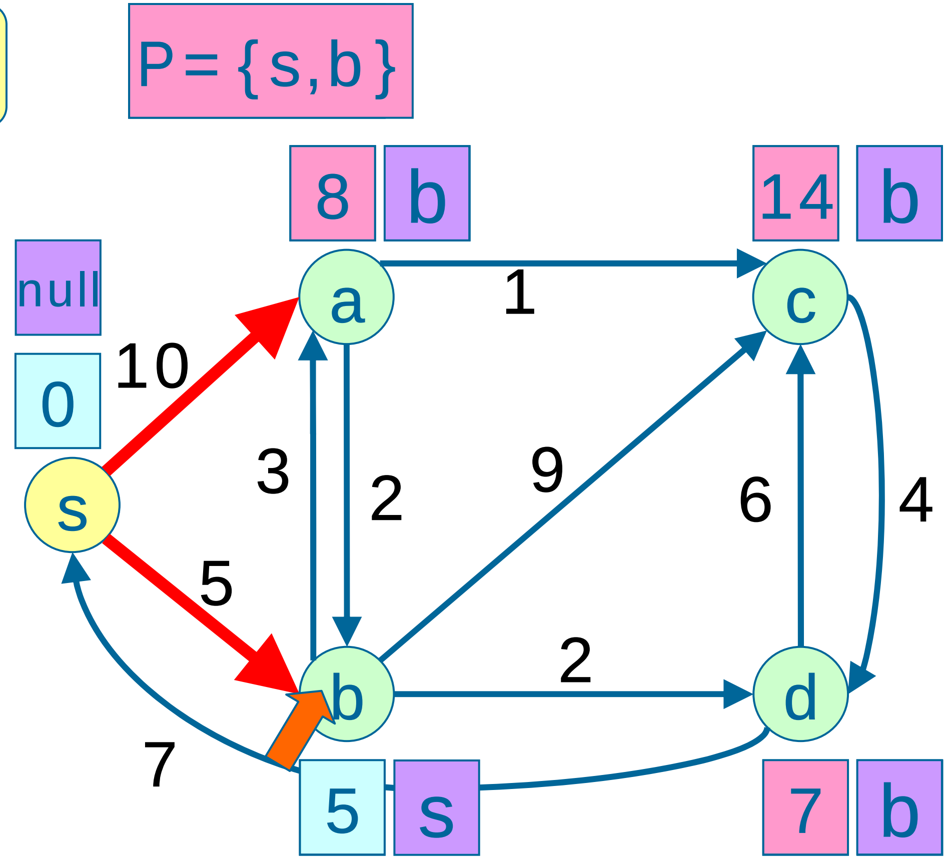


ダイクストラのアルゴリズム

- 全ての枝が非負の場合に，頂点 s から全ての頂点への最短路を求めるアルゴリズム

P=V となるまで反復

ステップ1：
d*(v)が最小な v∈V-Pを選ぶ
➔ u とおく
集合 P の更新: P := P∪{u}
d*(v), π(v) (v∈V-P)の更新
d*(v) > d*(u)+d(u,v)
ならば
d*(v) = d*(u)+d(u,v)
π(v) = u

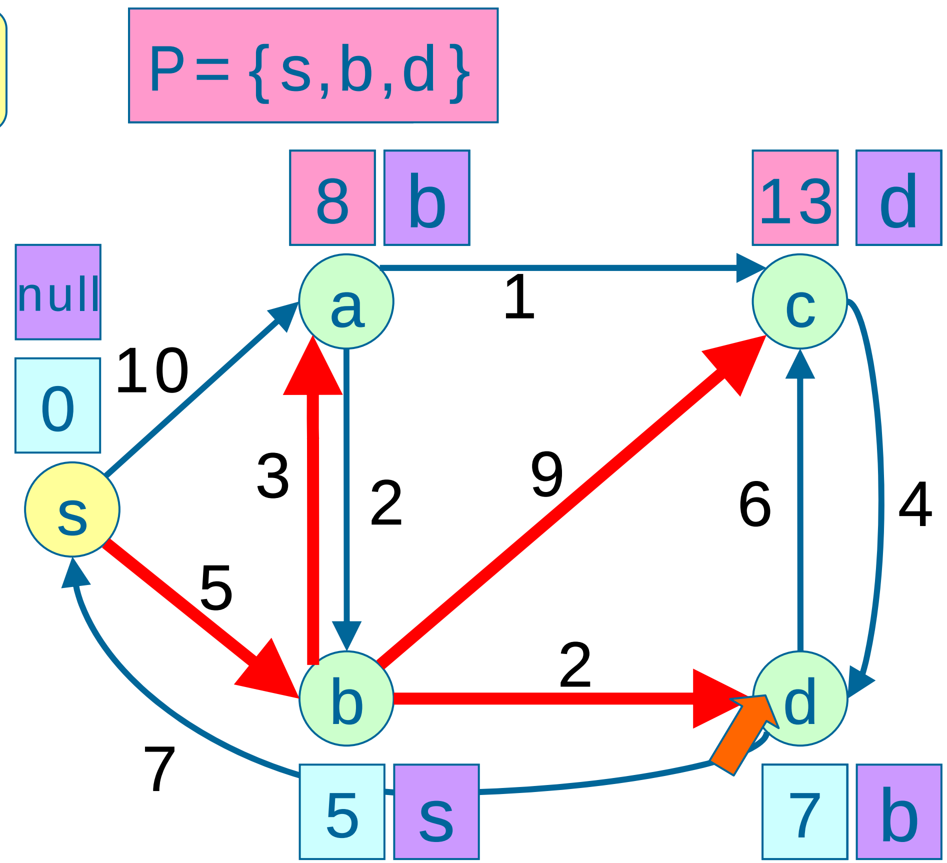


ダイクストラのアルゴリズム

- 全ての枝が非負の場合に，頂点 s から全ての頂点への最短路を求めるアルゴリズム

P=V となるまで反復

ステップ1:
d*(v)が最小な v∈V-Pを選ぶ
➔ u とおく
集合 P の更新: P := P∪{u}
d*(v), π(v) (v∈V-P)の更新
  d*(v) > d*(u)+d(u,v) ならば
    d*(v) = d*(u)+d(u,v)
    π(v) = u

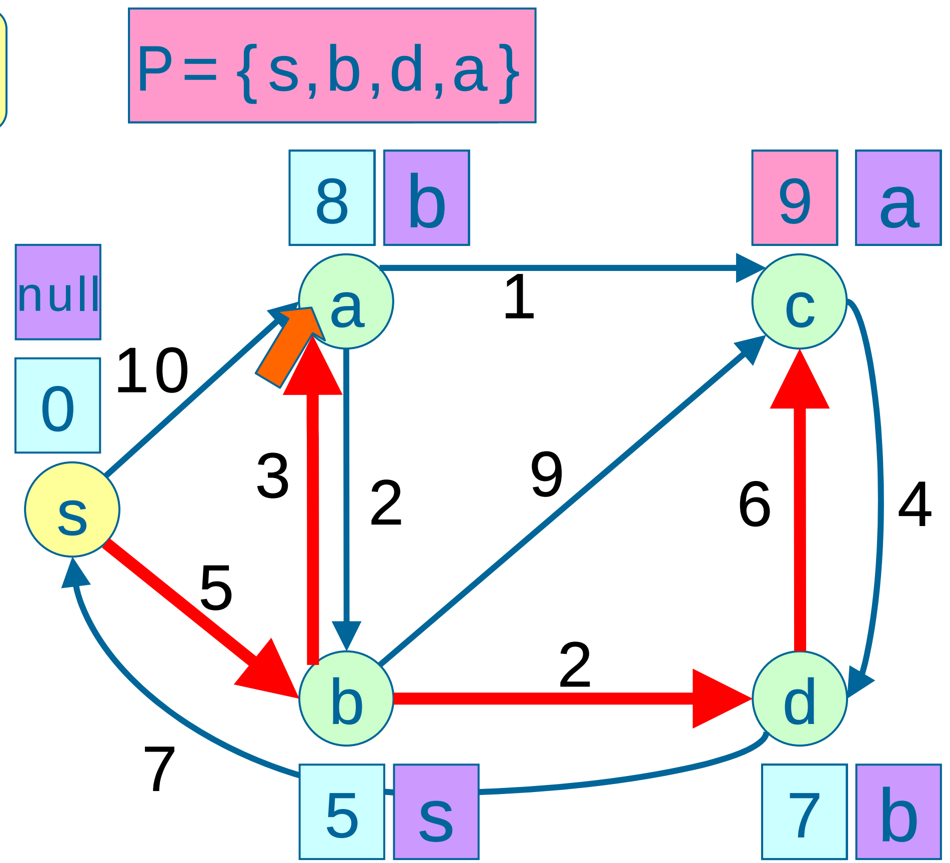


ダイクストラのアルゴリズム

- 全ての枝が非負の場合に, 頂点 s から全ての頂点への最短路を求めるアルゴリズム

ステップ1:
d*(v)が最小な v∈V-Pを選ぶ
➔ u とおく
集合 P の更新: P := P∪{u}
d*(v), π(v) (v∈V-P)の更新
  d*(v) > d*(u)+d(u,v) ならば
    d*(v) = d*(u)+d(u,v)
    π(v) = u

P=V となるまで反復

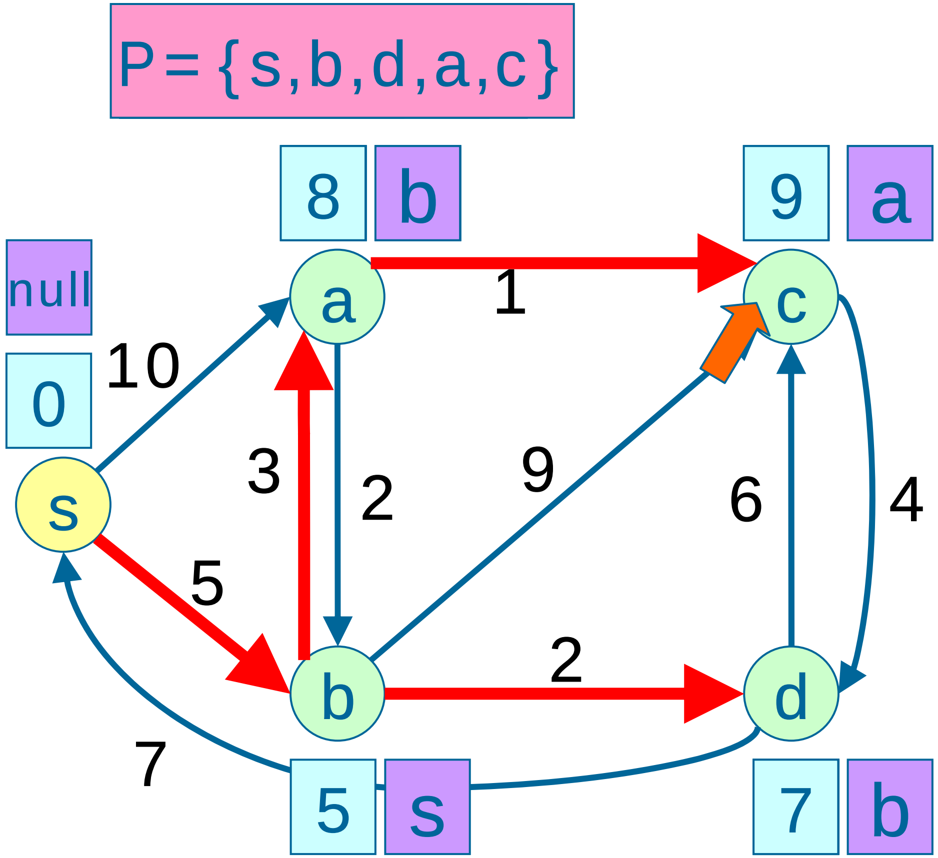


ダイクストラのアルゴリズム

- 全ての枝が非負の場合に，頂点 s から全ての頂点への最短路を求めるアルゴリズム

P=V となるまで反復

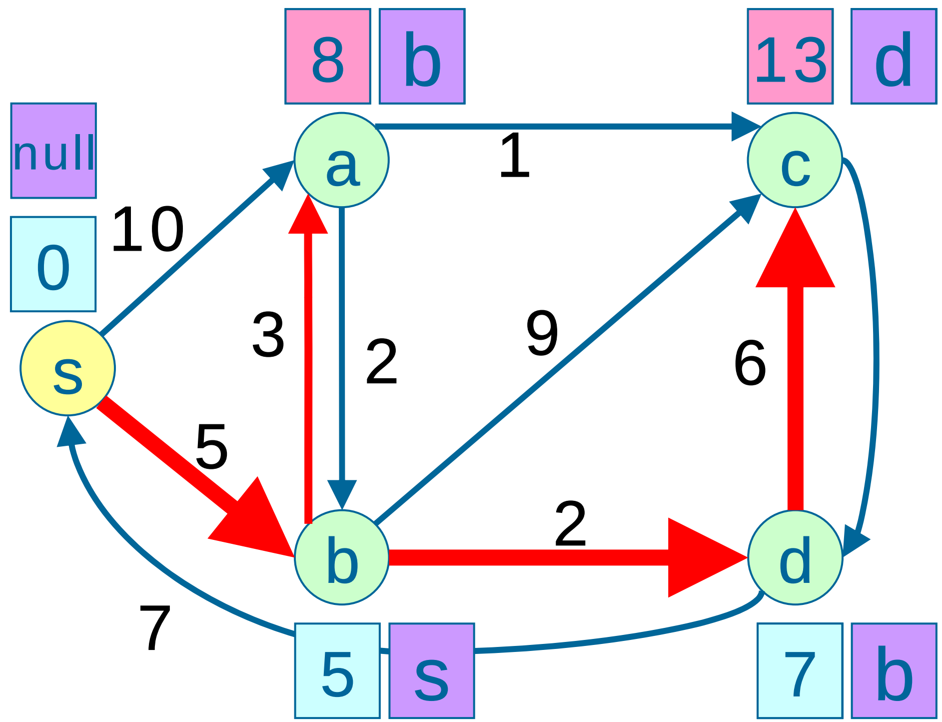
ステップ1:
d*(v)が最小な v∈V-Pを選ぶ
➔ u とおく
集合 P の更新: P := P∪{u}
d*(v), π(v) (v∈V-P)の更新
  d*(v) > d*(u)+d(u,v) ならば
    d*(v) = d*(u)+d(u,v)
    π(v) = u



d*(v)とπの性質

補題:ステップ1の各反復において, d*(v) < +∞ならば,
この値は s から v への路
　s→… →π(π(π(v))) →π(π(v)) →π(v) →v
の長さに等しい

4回目の反復開始時,
v=c の場合
π(c) = d, π(d) = b,
π(b) = s,
なので, d*(c) = 13 は
路s→b→d→c の長さに等しい



アルゴリズムの正当性

補題: ステップ1の各反復において，
P に含まれる頂点vの d*(v) の値は最短路長に等しい

[第1回目の反復]
第 1 回目の反復で P に追加された頂点は s
s への最短路長 ＝ 0 = d*(s)
よって成立

アルゴリズムの正当性

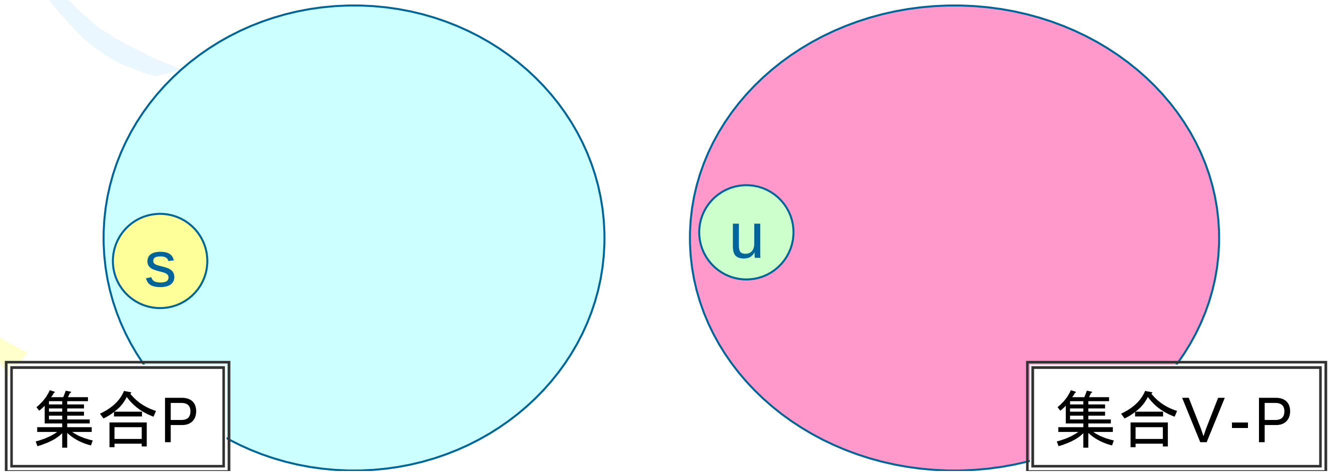
補題:ステップ1の各反復において，
P に含まれる頂点vの d*(v) の値は最短路長に等しい

[第k回目の反復]

u = 第 k 回目の反復で P に追加された頂点

➔第k回目の反復時において

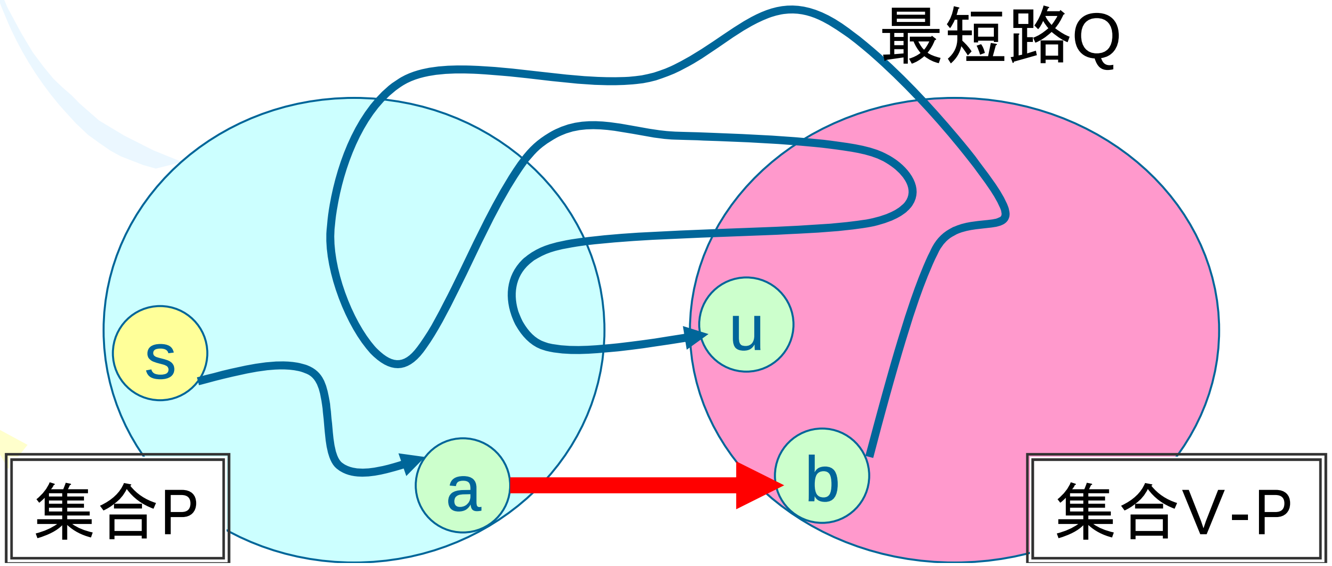
$$d^*(u) \leqq d^*(v) \quad (\forall v \in V-P)$$



アルゴリズムの正当性

補題:ステップ1の各反復において,
P に含まれる頂点vの d*(v) の値は最短路長に等しい

Q=s から u への最短路
枝(a,b): 路Q に沿って s から u に向かうときに,
最初に P から出る枝
※ b = u の可能性あり
d*(u) > (Q の長さ) と仮定, 矛盾を導く



アルゴリズムの正当性

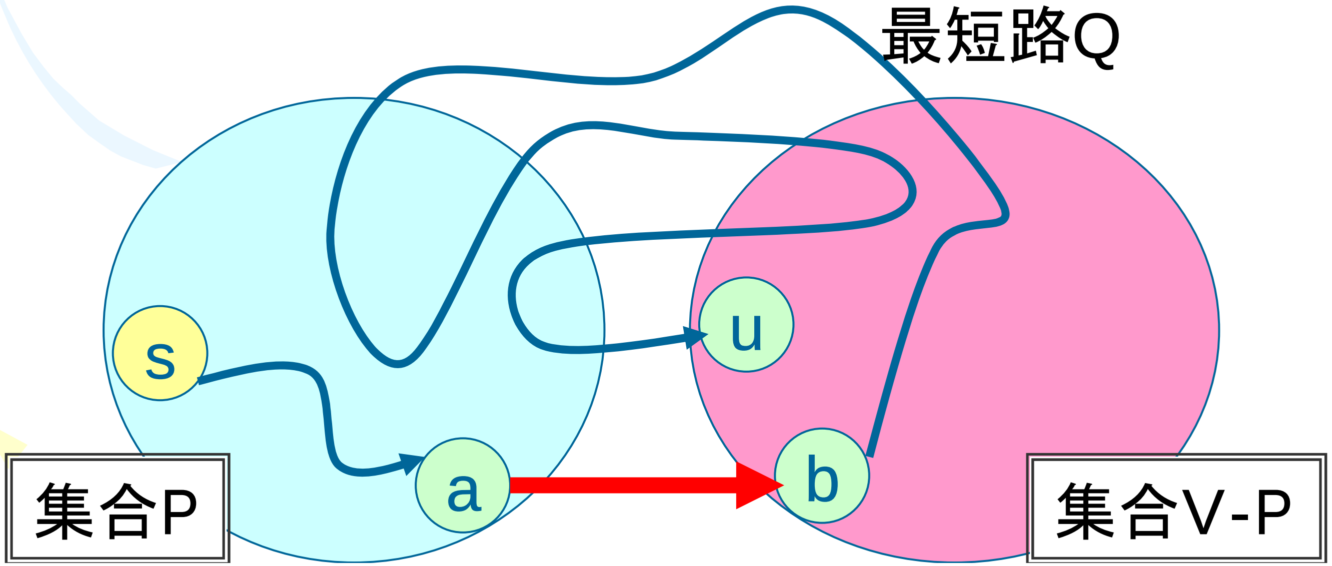
補題:ステップ1の各反復において，
P に含まれる頂点vの d*(v) の値は最短路長に等しい

頂点aは既にPに含まれている，b は V-P に含まれている

➔ d*(a) = sからaへの最短路長（帰納法の仮定より）

d*(b) ≦ d*(a) + d(a,b)

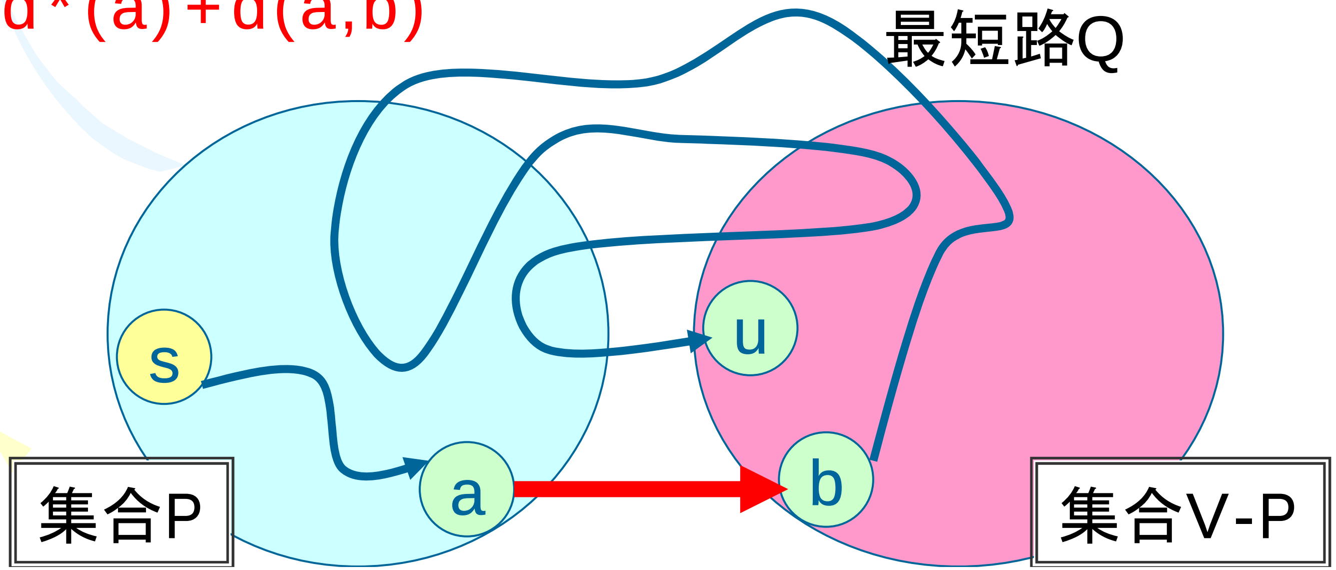
（aがPに追加されたとき，ステップ1が実行されたから）



アルゴリズムの正当性

補題:ステップ1の各反復において，
P に含まれる頂点vの d*(v) の値は最短路長に等しい

s から a へのQの部分路は最短路（Qの部分路だから）
➔ s から a へのQの部分路の長さ＝最短路長＝d*(a)
s から b へのQの部分路は最短路（Qの部分路だから）
➔ s から b へのQの部分路の長さ
= (s から a へのQの部分路の長さ) + d(a,b)
= d*(a)+d(a,b)



アルゴリズムの正当性

補題:ステップ1の各反復において,
P に含まれる頂点vの d*(v) の値は最短路長に等しい

まとめると,

d*(b) ≦ s から b へのQの部分路の長さ
≦ Qの長さ
< d*(u)

一方，b∈V-Pなので，d*(b)≧d*(u) （矛盾）

定理:アルゴリズム終了時に d*(v) (v∈V)の値は
s から v への最短路長に等しい

時間計算量の解析

- ステップ1でd*(v)を最小にする v を V-P から選ぶ
  --- ヒープを使う
  - ステップ0でのヒープの準備: O(m+n)時間
  - ステップ1でd*(v)を最小にする v を選び，取り出す:
    更新の時間 O(log n)，実行回数は n 回➔ O(n log n)
  - ステップ1でd*(v)を更新:
    更新の時間 O(log n)，実行回数は枝の本数 m 回
    ➔ O(m log n)

∴ 合計で O(m log n) 時間

レポート問題

下記のグラフに対して，次の2つの場合について
ダイクストラのアルゴリズムを用いて
全頂点への最短路を求めなさい．
(1) s を始点とした場合．(2) z を始点とした場合．
いずれの場合も，各反復において d*(v)，π，集合 P がどのように
変化していくかをきちんと書くこと．