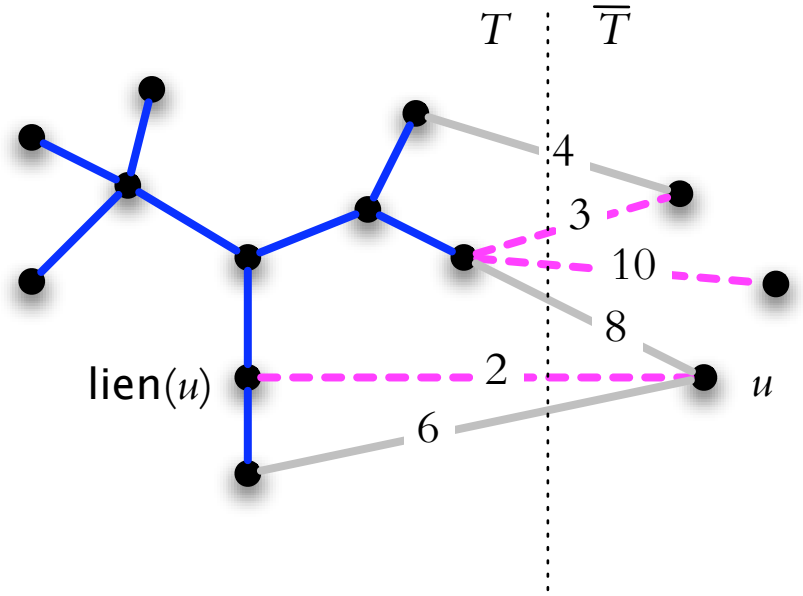
Prim



Idée de l'implantation : pour chaque $u \notin T$ la meilleure arête $(u, \mathsf{lien}(u))$ dans la coupure $(T, \bar{T})$ est placée dans un monceau

Prim : code

Implantation utilise un monceau H, on a besoin d'un sommet de départ s

```
      Algo ACM-PRIM(V, c, s)
AP1   pour tout u ∈ V initialiser cle(u) ← ∞ ; lien(u) ← null
AP2   cle(s) ← 0 ; initialiser H ← ∅ ; H.insert(s)
AP3   tandis que H ≠ ∅
AP4       u ← H.deleteMin() ; cle(u) ← −∞
AP5       ajouter arête (u, lien(u)) si lien(u) ≠ null
AP6       pour tout v ∈ Adj[u] faire
AP7           si c(u, v) < cle(v) alors
AP8               cle(v) ← c(u, v) ; lien(v) ← u
AP9               si v ∉ H alors H.insert(v) sinon H.decreaseKey(v)
```

Prim (cont)

$|V| = n$, $|E| = m$

n fois deleteMin(),
n fois insert(),
$m - n + 1$ fois decreaseKey().

Temps de calcul avec tas d-aire :

$O(nd \log_d n + m \log_d n)$, choisir $d = \lceil 2 + m/n \rceil$.
$\Rightarrow$ temps de $O(m \log_{2+m/n} n)$. Quand $m = \Omega(n^{1+\epsilon})$ avec $\epsilon > 0$, on a $O(m/\epsilon)$.

Temps de calcul avec tas Fibonacci :

$O(m + n \log n)$

Plus court chemins

Graphe $G = (V, E)$ orienté, arcs pondérés
(si graphe non-orienté, alors remplacer chaque arête par deux arcs aller-retour)

Rappel : poids d'un chemin $v_0, v_1, \ldots, v_\ell$ est $\sum_{i=1}^{\ell} c(v_{i-1}, v_i)$.

a trouver le plus court chemin de s à t
b trouver le plus court chemin de s à chaque sommet
c trouver le plus court chemin de chaque sommet à t
d trouver le plus court chemin entre chaque paire de sommets

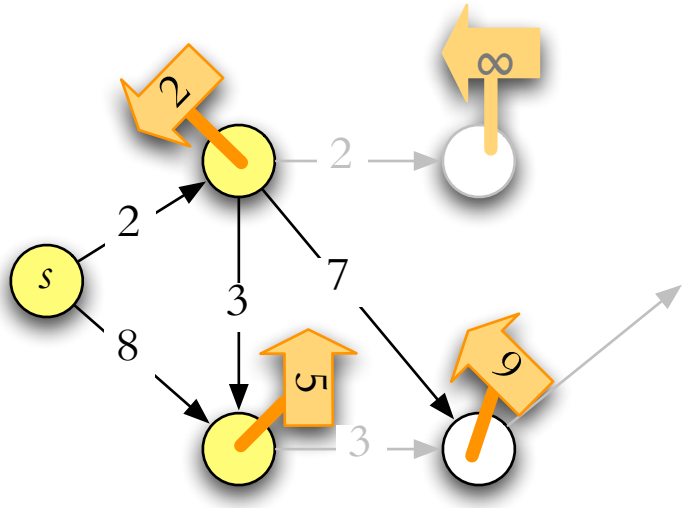
c équivaut **b** (renverser les arcs) ; toutes les solutions connues de **a** donnent une solution partielle à **b** ou **c**

Problème : cycle à poids négatif — le plus court chemin n'est pas défini

PCC - source

Plus court chemin d'une source s à chaque sommet

Idée d'étiquetage : maintenir un «panneau» à chaque sommet v : «il existe un chemin de longueur d à partir de sommet u»



Étiquetage : **si** $d(u)+c(u,v) < d(v)$ **alors** $d(v) \leftarrow d(u)+c(u,v)$; $p(v) \leftarrow u$

PCC - source

Amélioration : quand un sommet u est déjà étiqueté, on peut explorer tous les arcs partants — enregistrer que u est «exploré»

État d'un sommet : « non-étiqueté », « étiqueté » (mais pas exploré), ou « exploré »

Exploration : choisir un sommet u qui est «étiqueté», et faire

Procédure EXPLORER(u)
E1 pour tout $v \in \mathsf{Adj}[u]$
E2 **si** $d(u) + c(u,v) < d(v)$ **alors**
E3 $d(v) \leftarrow d(u) + c(u,v)$
E4 $p(v) \leftarrow u$
E5 changer l'état de v à «étiqueté»

PCC - exploration

Ordre d'exploration :

- dans le cas d'un graphe acyclique, utiliser le tri topologique
- s'il n'y a pas d'arcs à poids négatif, choisir le sommet étiqueté avec d minimum
  — implantation avec monceau
- s'il y a des arcs à poids négatif, choisir le sommet étiqueté le moins récemment
  — implantation avec queue

Algorithme de Dijkstra

```
     Algo PCC-DIJKSTRA(V, Adj[], c, s)
D1   pour tout u ∈ V initialiser d(u) ← ∞ ; p(u) ← null
D2   d(s) ← 0
D3   initialiser H ← ∅ ; H.insert(s) // (d est la clé dans H)
D4   tandis que H ≠ ∅ // (H est l'ensemble des sommets «étiquetés»)
D5       u ← H.deleteMin() ;
D6       pour tout v ∈ Adj[u] // (exploration de u)
D7           si d(u) + c(u, v) < d(v) alors
D8               d(v) ← d(u) + c(u, v) ; p(v) ← u
D9               si v ∉ H alors H.insert(v) sinon H.decreaseKey(v)
```

À la fin, $d(u)$ est la longueur du plus court chemin de s à chaque u, le chemin est donné en reculant $p(u), p(p(u)), \ldots$.

Dijkstra (cont)

Algorithme de Dijkstra (plus court chemin) est presque identique à celui de Prim (arbre couvrant minimal) !

Analyse comme avant : $|V| = n$, $|E| = m$

n fois deleteMin() (chaque sommet est exploré une fois),
n fois insert(),
$m - n + 1$ fois decreaseKey().

Temps de calcul avec tas d-aire :

$O(nd \log_d n + m \log_d n)$, choisir $d = \lceil 2 + m/n \rceil$.
$\Rightarrow$ temps de $O(m \log_{2+m/n} n)$. Quand $m = \Omega(n^{1+\epsilon})$ avec $\epsilon > 0$, on a $O(m/\epsilon)$.

Temps de calcul avec tas Fibonacci : $O(m + n \log n)$

Temps de calcul sans monceau : $O(n^2)$ [pour trouver $\min d$, il faut parcourir tous les sommets étiquetés n fois]

Bellman-Ford [+ Moore]

Et s'il y a des arcs à poids négatif ? (mais pas de cycles à poids négatif !)

Dijkstra ne trouve pas les plus court chemins

Il faut peut-être explorer un sommet plus qu'une fois — stocker les sommets dans une queue

Bellman-Ford (cont)

Algo PCC-BELLMAN$(V, \mathsf{Adj}[], c, s)$

B1 **pour tout** $u \in V$ initialiser $d(u) \leftarrow \infty$; $p(u) \leftarrow \mathsf{null}$

B2 $d(s) \leftarrow 0$

B3 initialiser $Q \leftarrow \emptyset$; $Q.\mathsf{enqueue}(s)$

B4 **tandis que** $Q \neq \emptyset$ // *(Q est l'ensemble des sommets «étiquetés»)*

B5 $\quad u \leftarrow Q.\mathsf{dequeue}()$;

B6 $\quad$ **pour tout** $v \in \mathsf{Adj}[u]$ // *(exploration de u)*

B7 $\quad\quad$ **si** $d(u) + c(u, v) < d(v)$ **alors**

B8 $\quad\quad\quad d(v) \leftarrow d(u) + c(u, v)$; $p(v) \leftarrow u$

B9 $\quad\quad\quad$ **si** $v \notin Q$ **alors** $Q.\mathsf{enqueue}(v)$

Notez qu'on doit tester si un sommet est sur la queue (une valeur booléenne stocké avec chaque sommet)

Bellman-Ford (cont)

Temps de calcul : $|V| = n$, $|E| = m$
définir les tournées $0, 1, 2, \ldots$

- en tournée 0, le sommet s est exploré
- en tournée $j > 0$, tous les sommets sont explorés qui sont dans la queue à la fin de tournée $(j - 1)$

Chaque tournée prend $O(m)$ temps

À la fin de tournée j, $d(u)$ est la longueur du plus court chemin pour chaque u t.q. le chemin $s \rightsquigarrow u$ contient j arcs

Si la queue n'est pas vide à la fin de tournée $(n - 1)$, alors il y a un cycle à poids négatif et l'algorithme ne finit jamais