

Approximation

(suite et fin)

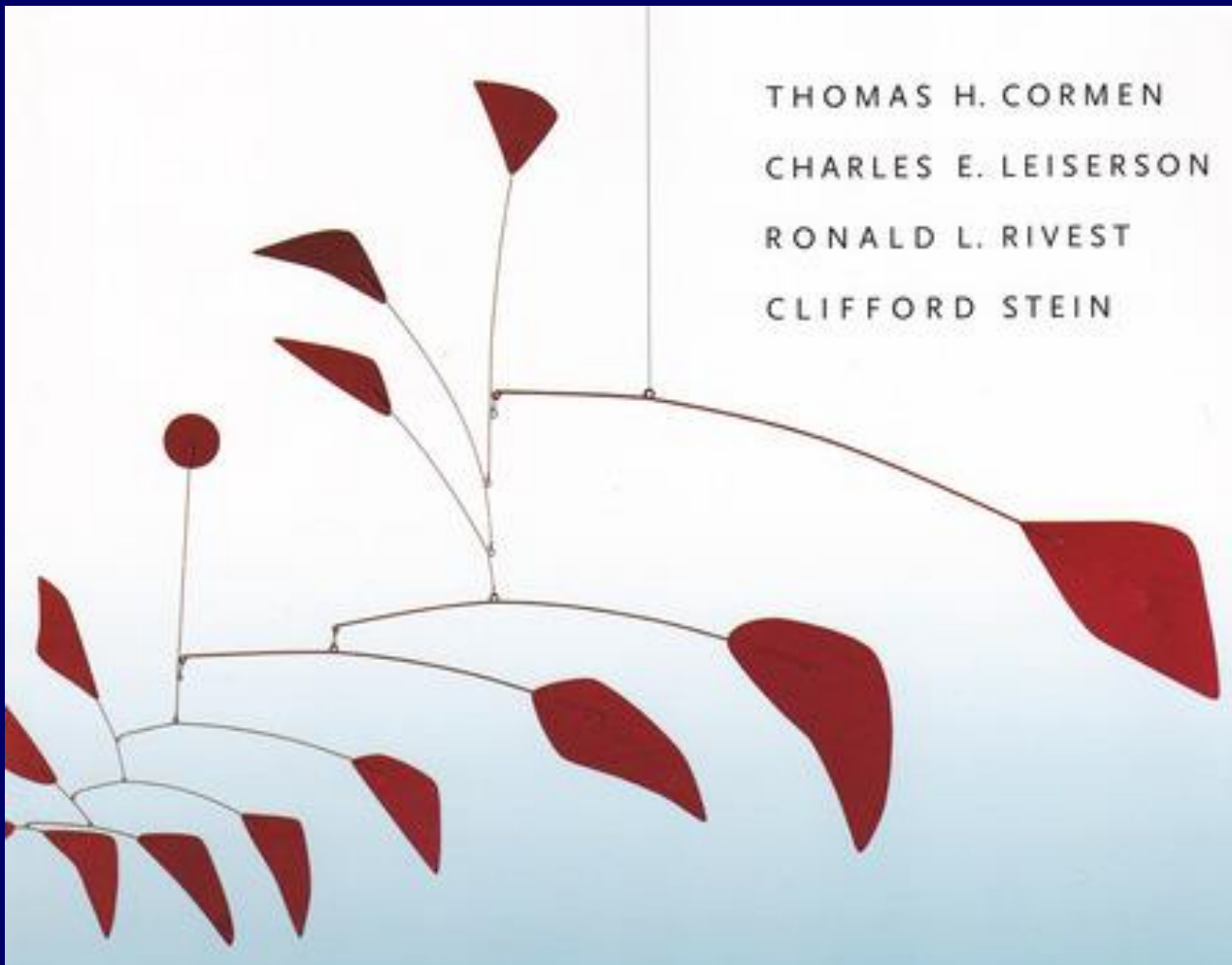
Le LIVRE

Pour toute la partie Algorithmique :

**Thomas H. Cormen, Charles E. Leiserson,
Ronald L. Rivest : Introduction to algorithms,
MIT Press, 1990.**

La dernière édition :

**Thomas H. Cormen, Charles E. Leiserson,
Ronald L. Rivest, Clifford Stein : Introduction
to algorithms, MIT Press, 2009.**



THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN

INTRODUCTION TO

ALGORITHMS

THIRD EDITION

VF

**Thomas H. Cormen, Charles E. Leiserson,
Ronald L. Rivest, Clifford Stein : Introduction
à l'algorithmique, *troisième édition*, Dunod, juin
2010.**

**Cormen • Leiserson
Rivest • Stein**

Cours, exercices et problèmes

Algorithmique

**Cours avec 957 exercices
et 158 problèmes**

Plus de
20 000 exemplaires
vendus

3^e édition
avec compléments en ligne

DUNOD

Approximation de SSP

Le problème de décision : étant donné l'ensemble de nombres naturels $S=\{x_1, x_2, \dots, x_n\}$ et un nombre naturel t , on pose la question s'il existe un sous-ensemble de S , dont la somme des éléments soit t .

Le problème d'optimisation : étant donné l'ensemble de nombres naturels $S=\{x_1, x_2, \dots, x_n\}$ et un nombre naturel t , on cherche le plus grand $t' \leq t$, tel qu'il existe un sous-ensemble de S , dont la somme des éléments soit t' .

Notation

- on notera " $L + x$ ", la liste triée obtenue par addition de x à chaque élément de la liste triée L
- on notera " $\text{merge}(L, L')$ " la liste triée et sans doublons obtenue à partir des listes triées L et L'
- on notera " $\text{supr}(L, k)$ " la liste triée obtenue à partir de la liste triée L par suppression des éléments supérieurs à k

Algo exponentiel pour SSP

$n \leftarrow |S|$

$L_0 \leftarrow \langle 0 \rangle$

pour $i = 1$ à n **faire**

$L_i \leftarrow \text{merge}(L_{i-1}, L_{i-1} + x_i)$

$L_i \leftarrow \text{supr}(L_i, t)$

return(maximum de L_n)

Example

$S=\{1, 4, 7, 10\}$

$t=20$

$L_0=\{0\}$

$L_1=\{0, 1\}$

$L_2=\{0, 1, 4, 5\}$

$L_3=\{0, 1, 4, 5, 7, 8, 11, 12\}$

$L_4=\{0, 1, 4, 5, 7, 8, 10, 11, 12, 14, 15, 17, 18\}$

Résultat : 18

Le même exemple encore

$S=\{10, 7, 4, 1\}$

$t=20$

$L_0=\{0\}$

$L_1=\{0, 10\}$

$L_2=\{0, 7, 10, 17\}$

$L_3=\{0, 4, 7, 10, 11, 14, 17\}$

$L_4=\{0, 1, 4, 5, 7, 8, 10, 11, 12, 14, 15, 17, 18\}$

Résultat : 18

Exemple de l'exponentialité

$$S = \{1, 2, 4, 8, 16, \dots\} = \{2^{i-1} \mid i=1, \dots, n\}$$

t assez grand $\Theta(2^n)$

$$L_1 = \{0, 1\}$$

$$L_2 = \{0, 1, 2, 3\}$$

...

$$L_i = \{0, 1, \dots, 2^{i-1}\}$$

et comme la taille des L_i est exponentielle,
l'algorithme l'est.

Un FPTAS pour SSP

L'idée est d'utiliser un procédé d'*élagage*.

Soit $0 < \delta < 1$. Un *élagage* de la liste L par δ , consiste en la suppression de maximum d'éléments de L , de manière à obtenir une liste L' , qui pour tout élément supprimé y contient un élément proche, c.a.d. $z \leq y$, tel que $(y-z)/y \leq \delta$

$$\text{c.a.d. } (1-\delta)y \leq z \leq y$$

On peut considérer que z représente y ; ainsi chaque valeur est soit présente soit représentée par une valeur assez proche.

Exemple d'élagage

$L = \{10, 11, 12, 15, 20, 21, 22, 23, 24, 29\}$

$\delta = 1/10$

$L = \{10, \textcolor{red}{11}, 12, 15, 20, \textcolor{red}{21}, \textcolor{red}{22}, 23, \textcolor{red}{24}, 29\}$

11 représenté par 10

21 et 22 représentés par 20

24 représenté par 23

$L' = \{10, 12, 15, 20, 23, 29\}$

Algorithme d'élagage en $O(m)$

ELAGAGE(L, δ)

$m \leftarrow |L|$

$L' \leftarrow \{y_1\}$

$\text{last} \leftarrow y_1$

pour $i = 2$ à m **faire**

si $\text{last} < (1 - \delta) y_i$

alors

$L' \leftarrow L' \leftarrow \{y_i\}$

$\text{last} \leftarrow y_i$

reurn(L')

Le schéma d'approximation

Approx SSP(S, t, ε)

$n \leftarrow |S|$

$L_0 \leftarrow \{0\}$

pour $i = 1$ à n faire

$L_i \leftarrow \text{merge}(L_{i-1}, L_{i-1} + x_i)$

$L_i \leftarrow \text{elagage}(L_i, \varepsilon/n)$

$L_i \leftarrow \text{supr}(L_i, t)$

return(maximum de L_n)

Exemple

$$S = \{24, 28, 36, 32\}$$

$$t = 66$$

$$\varepsilon = 0,3 \Rightarrow$$

$$\delta = \varepsilon/4 = 0,075$$

$$L_0 = \{0\}$$

$$M \quad L_1 = \{0, 24\}$$

$$E \quad L_1 = \{0, 24\}$$

$$S \quad L_1 = \{0, 24\}$$

$$M \quad L_2 = \{0, 24, 28, 52\}$$

$$E \quad L_2 = \{0, 24, 28, 52\}$$

$$S \quad L_2 = \{0, 24, 28, 52\}$$

$$M \quad L_3 = \{0, 24, 28, 36, 52, 60, 64, 88\}$$

$$E \quad L_3 = \{0, 24, 28, 36, 52, 60, 88\}$$

$$S \quad L_3 = \{0, 24, 28, 36, 52, 60\}$$

$$M \quad L_4 = \{0, 24, 28, 32, 36, 52, 56, 60, 68, 84, 88, 92\}$$

$$E \quad L_4 = \{0, 24, 28, 32, 36, 52, 60, 68, 84, 88, 92\}$$

$$S \quad L_4 = \{0, 24, 28, 32, 36, 52, 60\}$$

Résultat : 60

(Le résultat optimal est 64)

Preuve de l'algorithme

- Dans toutes les opérations on manipule des sommes d'éléments (et éventuellement on en supprime) donc le résultat est la somme de certains éléments de S.
- Reste à montrer que
$$\text{Résultat} \geq (1 - \varepsilon) \text{ Optimum}$$
- Il faut montrer que la complexité de l'algorithme est polynomiale.

L'approximation

Un élément élagué, y est représenté par un z tel que :

$$(1 - \varepsilon/n) y \leq z \leq y$$

Lors d'une phase ultérieure, on obtient

$$(1 - \varepsilon/n)^i y \leq z \leq y$$

Ainsi, pour chaque élément dans L_i , y est soit présent soit représenté par un z tel que

$$(1 - \varepsilon/n)^i y \leq z \leq y$$

Donc le résultat vérifie

$$(1 - \varepsilon/n)^n Opt \leq Rés \leq Opt$$

Approximation (suite)

Mais comme la fonction

$$f(n) = (1 - \varepsilon/n)^n$$

est croissante en n (dérivée positive !), on a

$$(1 - \varepsilon) < (1 - \varepsilon/n)^n$$

Donc

$$(1 - \varepsilon) \textit{Opt} \leq \textit{Rés} \leq \textit{Opt}$$

La complexité

Après l'élagage, si z et z' sont des éléments consécutifs dans L_i , alors on ne peut pas avoir

$$z' (1 - \varepsilon/n) < z < z'$$

donc on a

$$z' (1 - \varepsilon/n) > z$$

c.a.d.

$$z'/z > 1/(1 - \varepsilon/n)$$

Ainsi le nombre d'éléments de L_i est au plus

$$\log_{1/(1 - \varepsilon/n)} t = \ln t / (-\ln(1 - \varepsilon/n)) \leq n \ln t / \varepsilon$$

Complexité (suite)

Le nombre d'éléments de L_i est donc polynomial en n , $\ln t$ et $1/\varepsilon$.

La première valeur est bornée par la taille des données, la seconde est la taille de la représentation de t et la troisième ($1/\varepsilon$) est celle demandée par un FPTAS.

CQFD

Une dernière remarque concernant la NP-complétude

Problèmes de reconnaissance vs optimisation

Si on sait reconnaître en temps polynomial, alors en un nombre $O(\log Rés)$ de reconnaissances on peut trouver l'optimum.

Si on sait trouver l'optimum alors on sait répondre au problème de reconnaissance.

Remarque : pour les problèmes d'optimisation on parle de NP-difficulté seulement.

FLOTS

Définitions

Un réseau est un graphe orienté $G(V,E)$ dans lequel chaque arc (u,v) dispose d'une capacité $c(u,v) \geq 0$.

Un réseau est doté de deux nœuds spéciaux :

s – source

t – puits

Nous supposons par la suite que pour tout sommet x du réseau il existe un chemin de s vers x et un de x vers t .

Définitions (suite)

Un flot dans G est une fonction $f : E \rightarrow \mathbb{R}$ qui vérifie :

- contrainte de capacité

$$\forall e \in E : 0 \leq f(e) \leq c(e)$$

- conservation des flots (lois de Kirchhoff)

Soit $\text{In}(v)$ l'ensemble des arcs entrants en v et $\text{Out}(v)$ l'ensemble des arcs sortants de v . Alors

$$\forall v \in V - \{s, t\} : \sum_{e \in \text{In}(v)} f(e) - \sum_{e \in \text{Out}(v)} f(e) = 0$$

Définitions (suite)

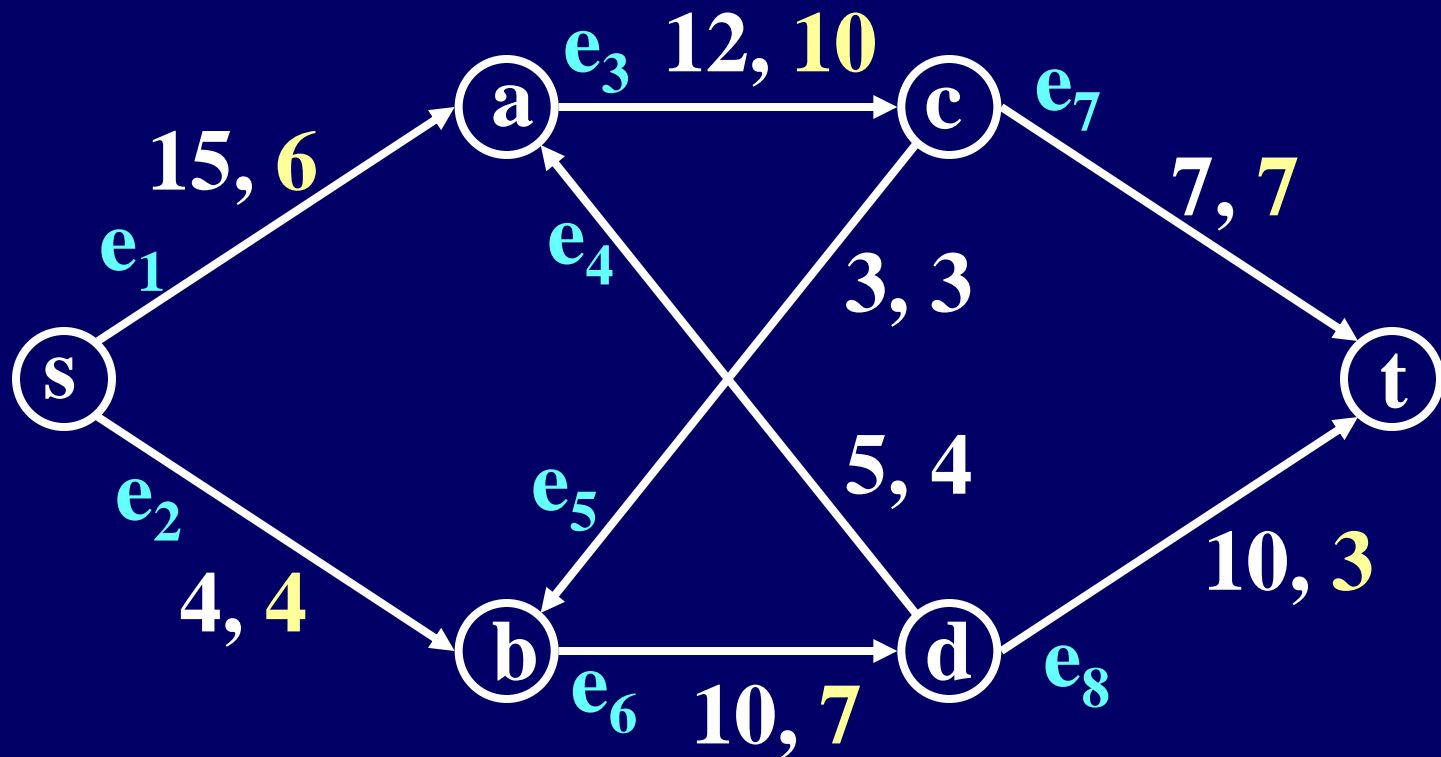
La valeur d'un flot

$$|f| = \sum_{e \in \text{In}(t)} f(e) - \sum_{e \in \text{Out}(t)} f(e)$$

Le problème du flot maximum :

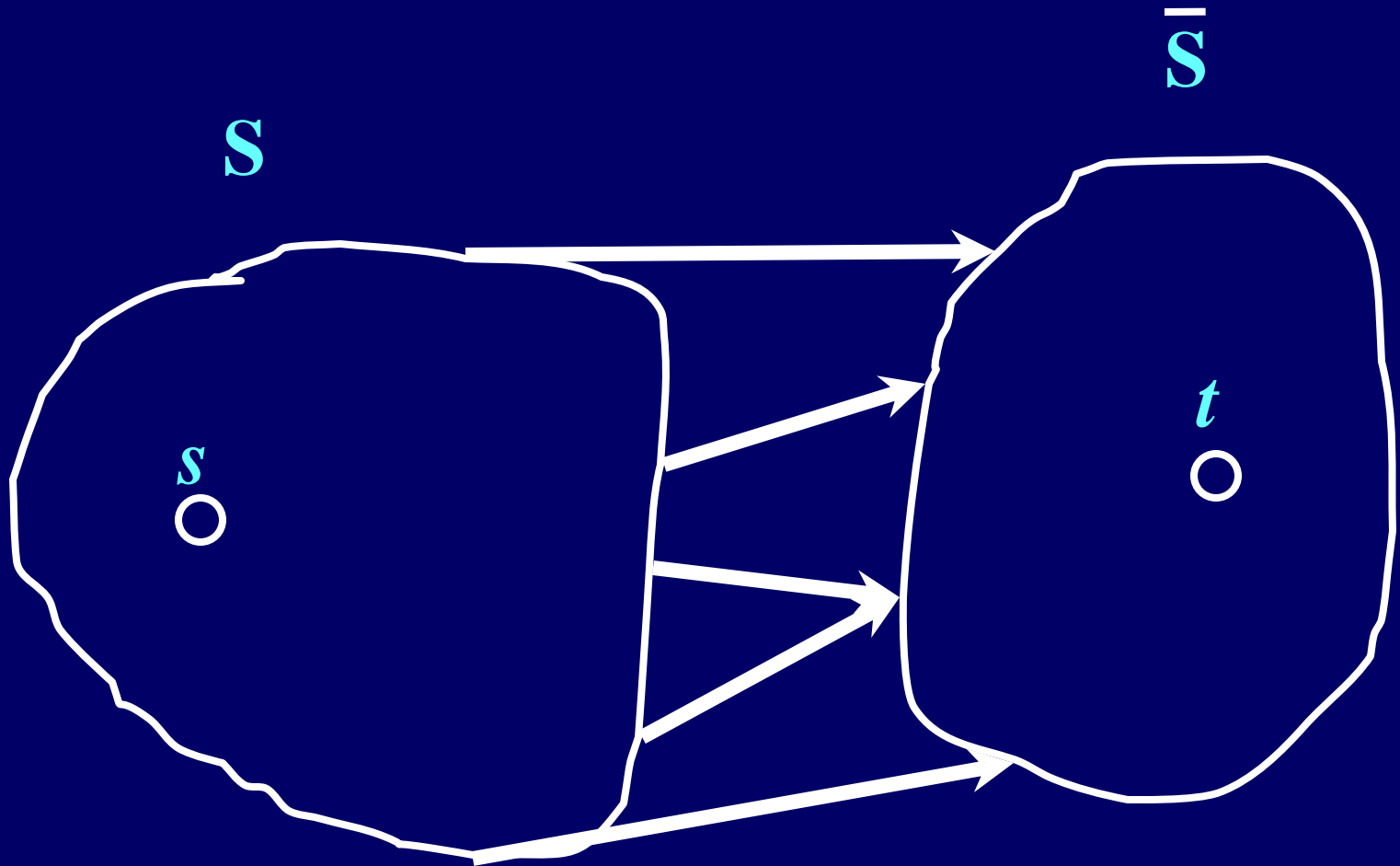
trouver un flot de valeur maximum.

Example



c, f

Une coupe



Definition

Soient $S, T \subset V$. (S,T) désigne l'ensemble des arcs ayant l'origine dans S et l'extrémité terminale dans T .

Soit $S \subset V$, t.q. $s \in S$ et $t \notin S$. L'ensemble des arcs entre S et son complémentaire S' s'appelle une coupe (ou une s - t coupe). Il est noté $(S;S')$

Une propriété

Propriété : Pour tout s-t coupe (S, S') ,

$$|f| = \sum_{e \in (S; S')} f(e) - \sum_{e \in (S'; S)} f(e)$$

Preuve : Il suffit de sommer les équations

$$\sum_{e \in \text{In}(v)} f(e) - \sum_{e \in \text{Out}(v)} f(e) = 0$$

pour les sommets $v \in S'$ ($v \neq t$)

et

$$|f| = \sum_{e \in \text{In}(t)} f(e) - \sum_{e \in \text{Out}(t)} f(e)$$

La capacité d'une coupe

On définit

$$c(S) = \sum_{e \in (S; S')} c(e)$$

Une borne

Propriété : pour tout flot f et pour toute coupe S

$$|f| \leq c(S)$$

Preuve : par la propriété précédente nous avons

$$|f| = \sum_{e \in (S; S')} f(e) - \sum_{e \in (S'; S)} f(e)$$

et par ailleurs $0 \leq f(e) \leq c(e)$ pour toute arête e .

Donc

$$|f| \leq \sum_{e \in (S; S')} c(e) - 0 = c(S)$$

Un corollaire

Corollaire : Si $|f| = c(S)$ alors le flot est maximum et la coupe S est de capacité minimum.

Chaîne augmentante

Une chaîne augmentante est un chaîne entre s et t sur laquelle on peut augmenter le flot,

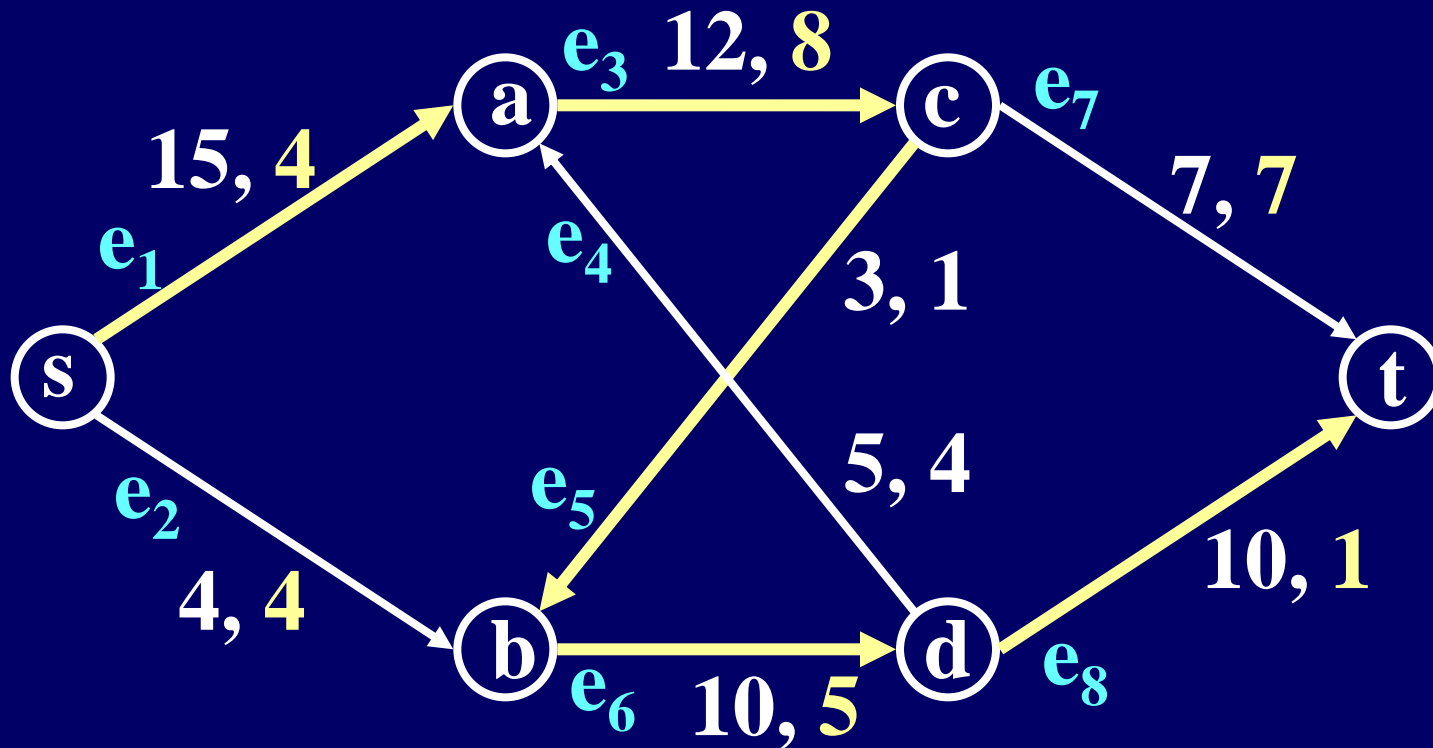
c.a.d. une chaîne sur laquelle les arcs "avant" vérifient

$$f(e) < c(e)$$

et les arcs "arrière" vérifient

$$f(e) > 0$$

Chaîne augmentante - exemple



c, f

Méthode Ford & Fulkerson

**tant qu'il existe une chaîne augmentante
faire**

- **chercher une chaîne augmentante**
- **augmenter le flot sur la chaîne**

fait

**A préciser : comment chercher, de combien
augmenter**

Réseau résiduel

Si on a un arc de u vers v , alors

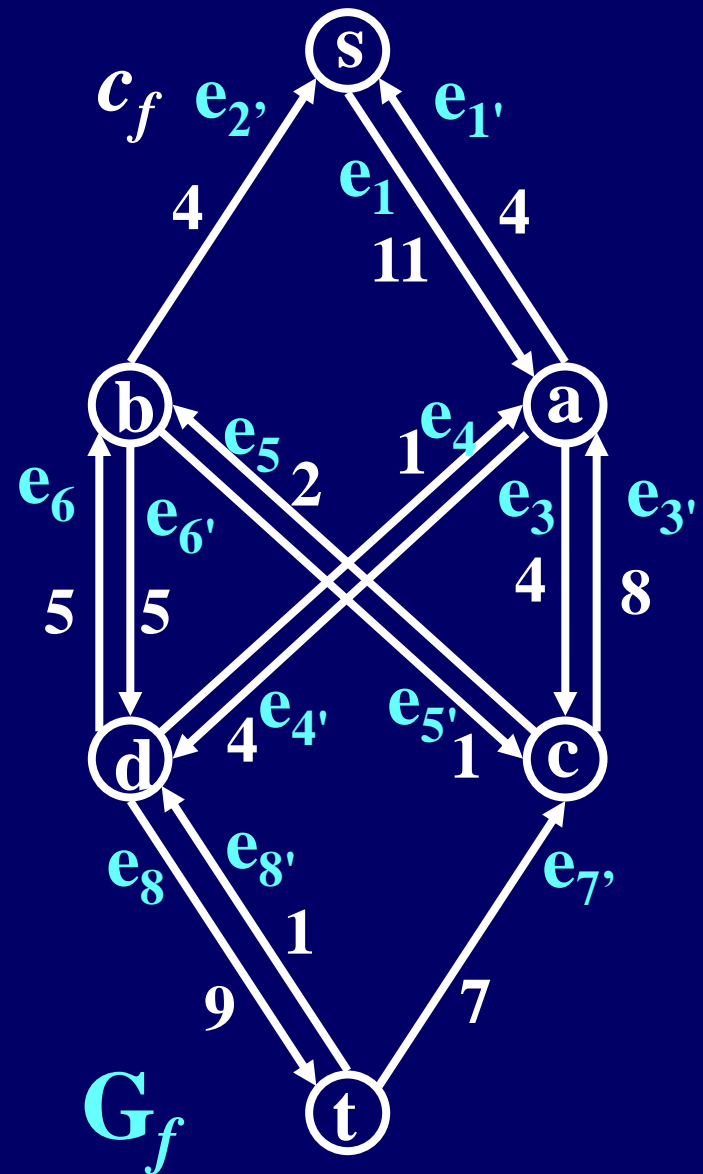
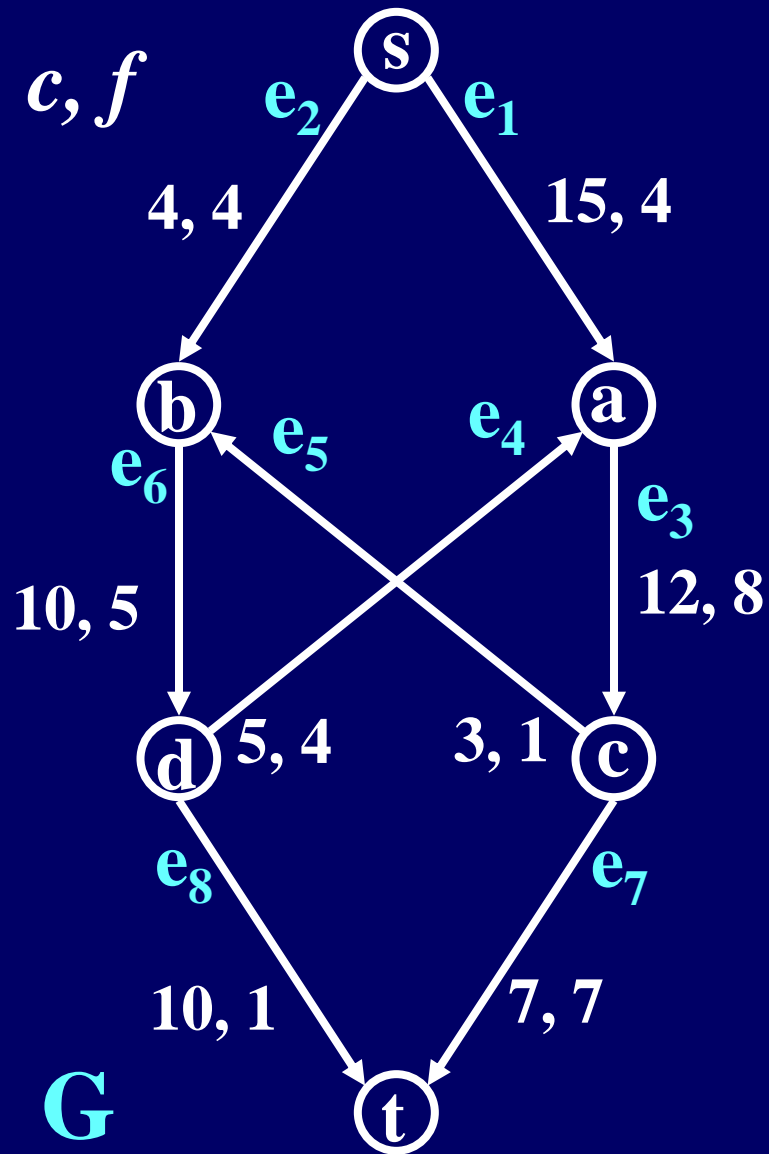
on met dans G_f un arc de u vers v de capacité

$$c_f(u,v) = c(u,v) - f(u,v)$$

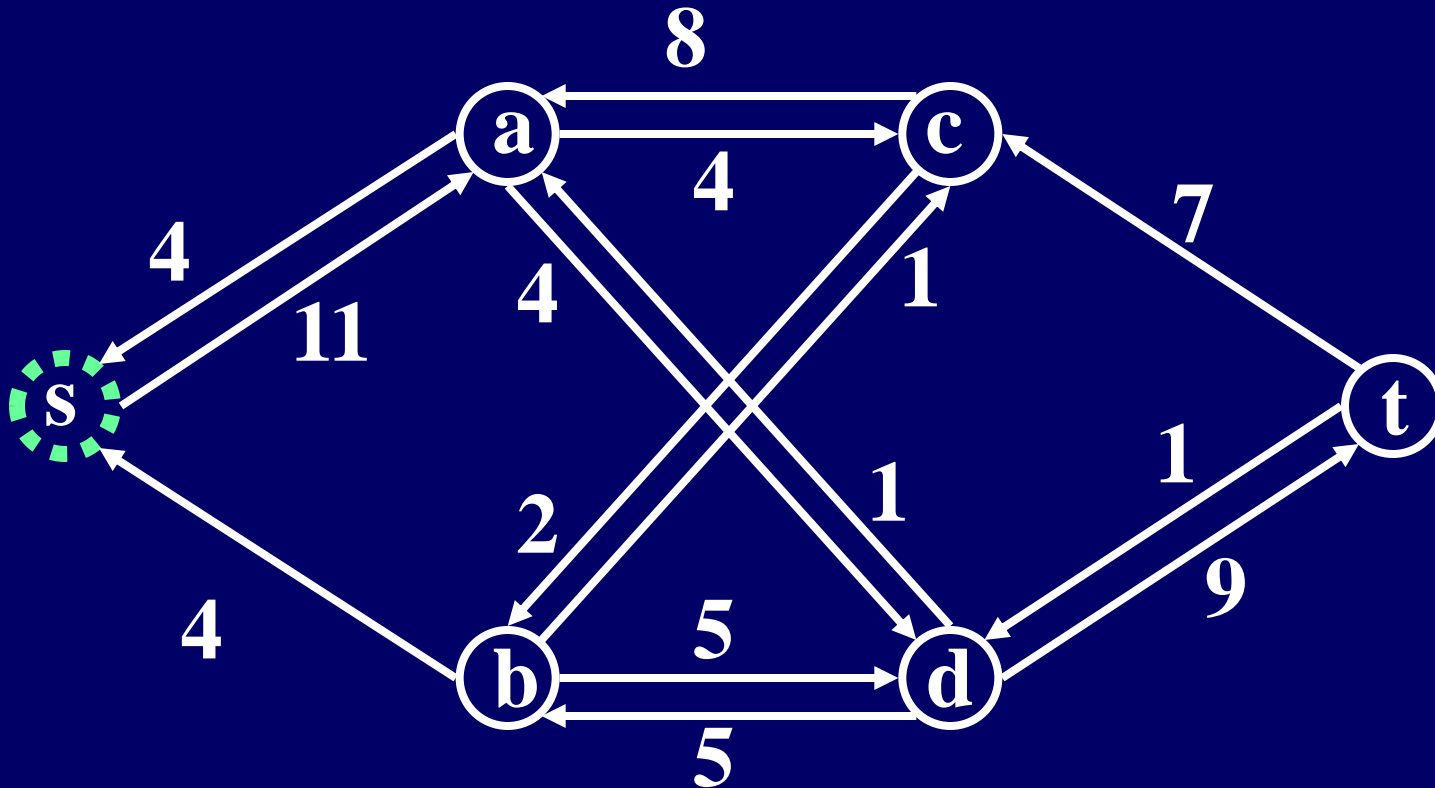
on met dans G_f un arc de v vers u de capacité

$$c_f(v,u) = f(u,v)$$

Le réseau résiduel

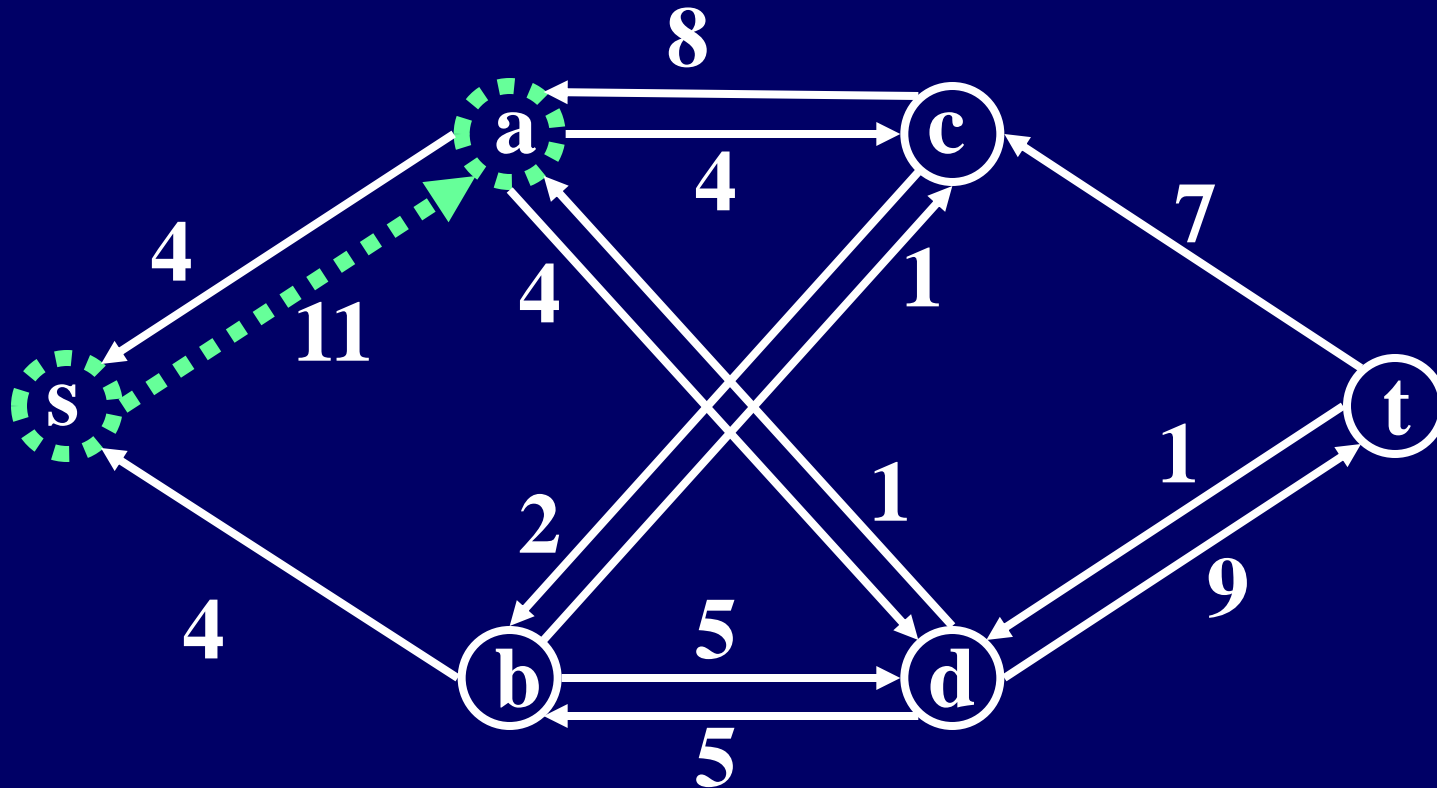


Recherche de chaîne augmentante



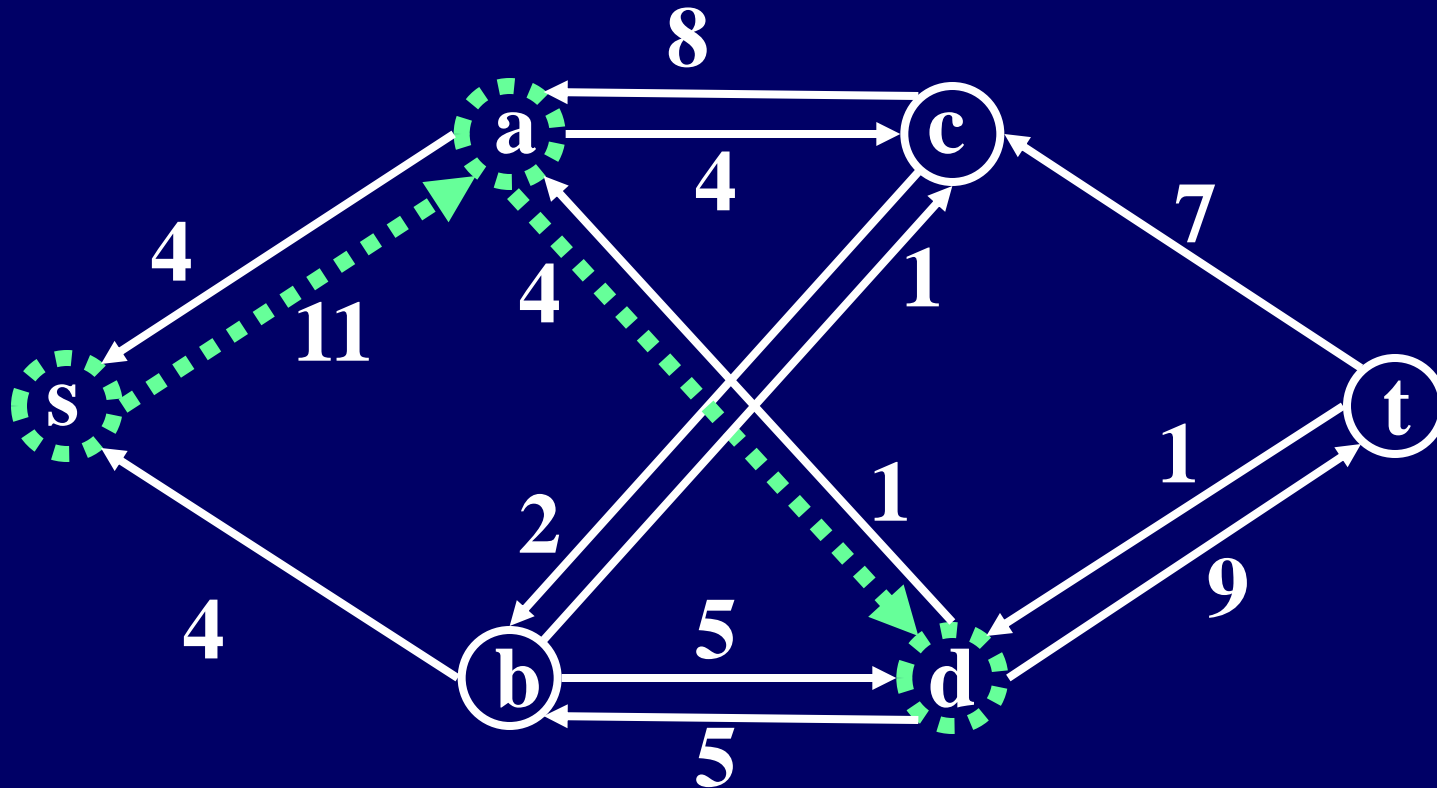
c_f

Recherche de chaîne augmentante



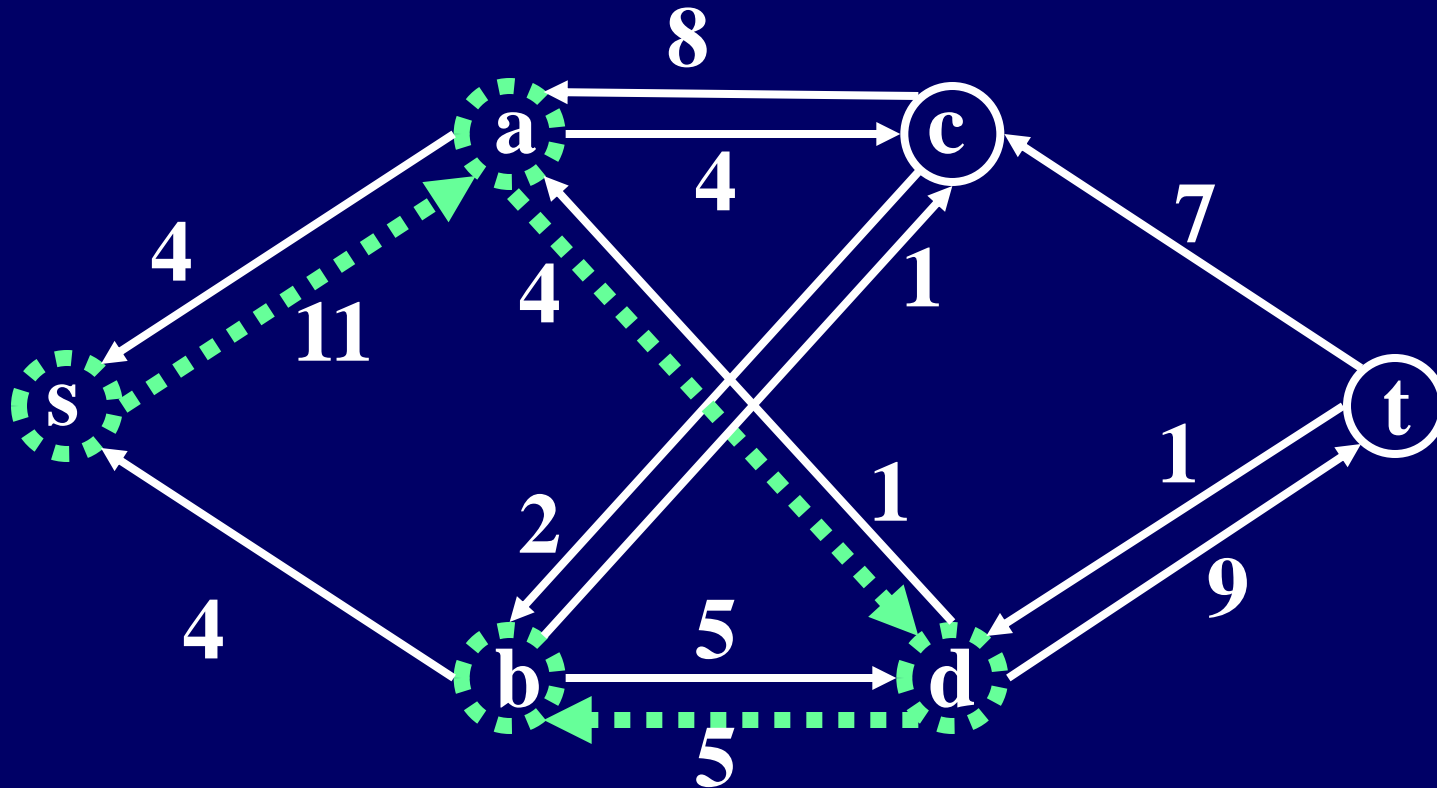
c_f

Recherche de chaîne augmentante



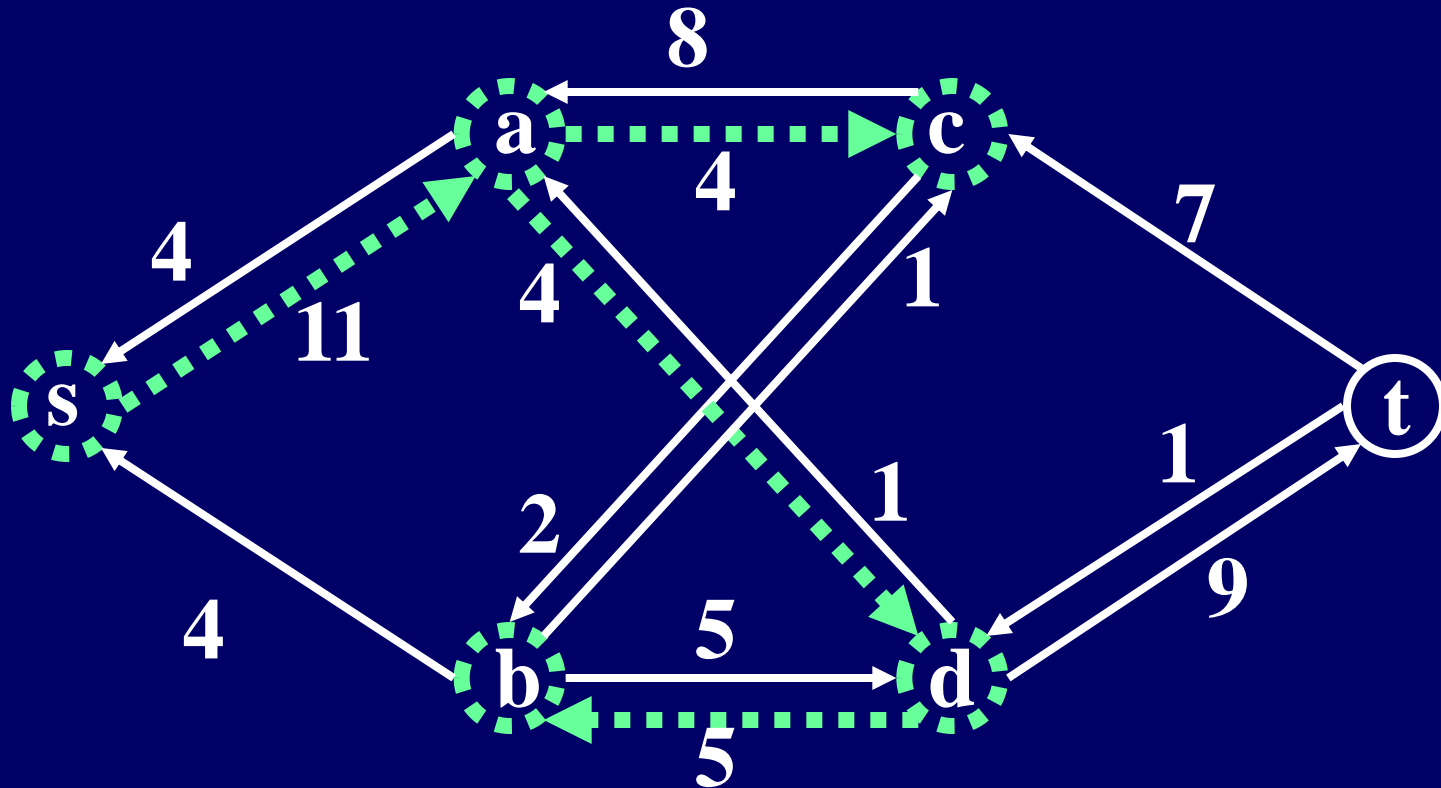
c_f

Recherche de chaîne augmentante



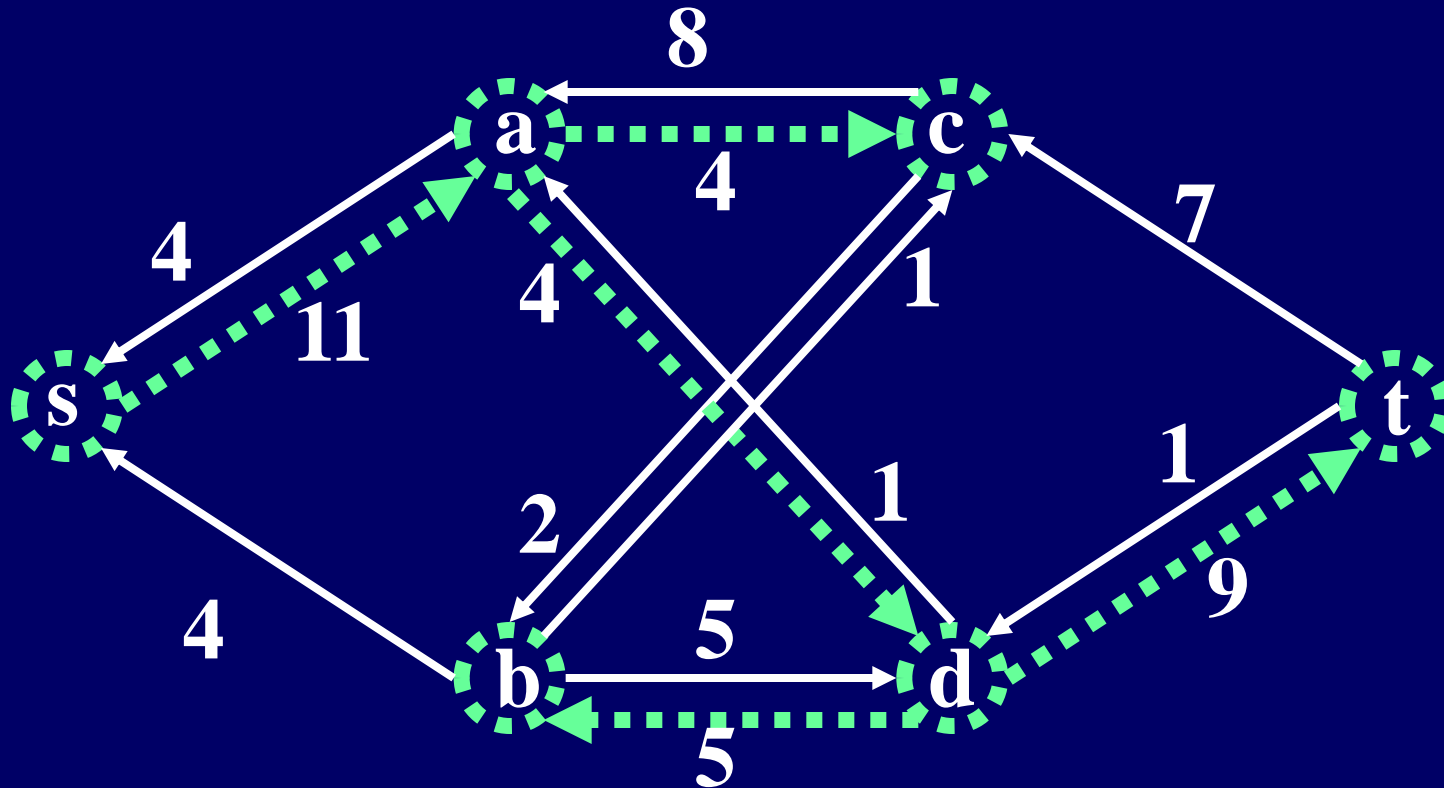
c_f

Recherche de chaîne augmentante



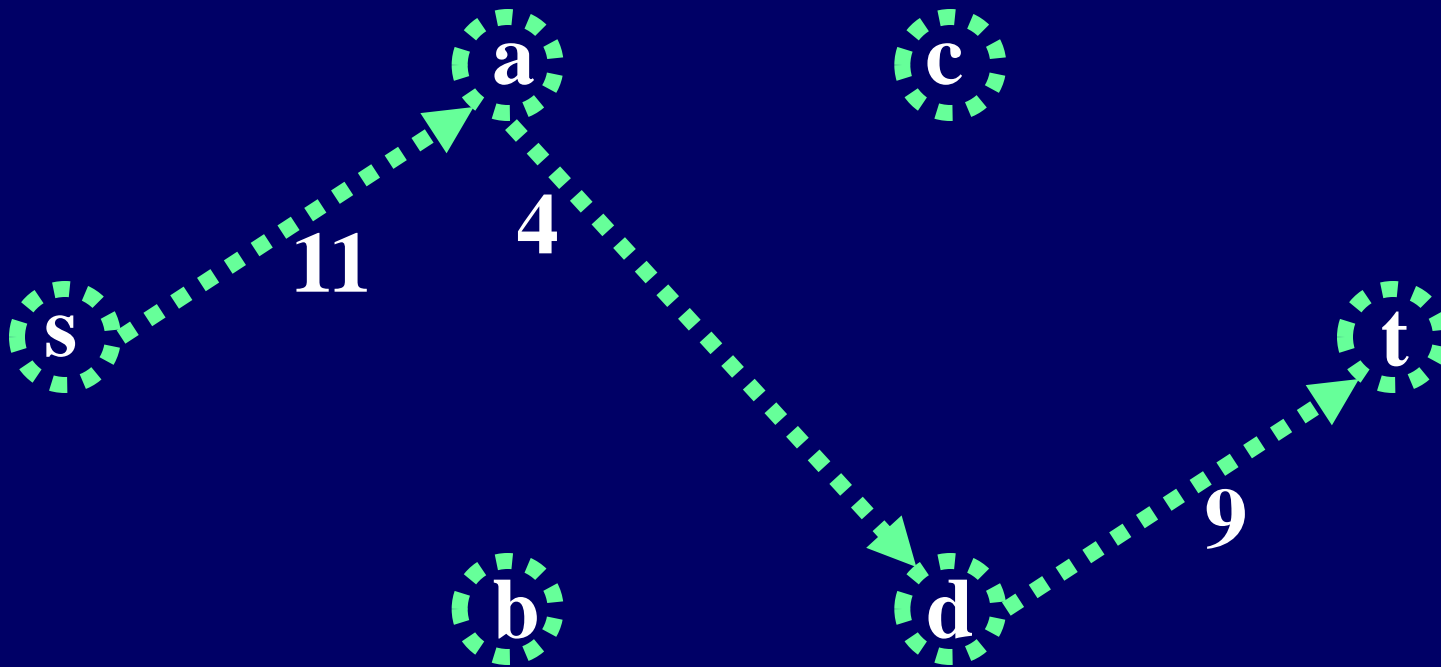
c_f

Recherche de chaîne augmentante



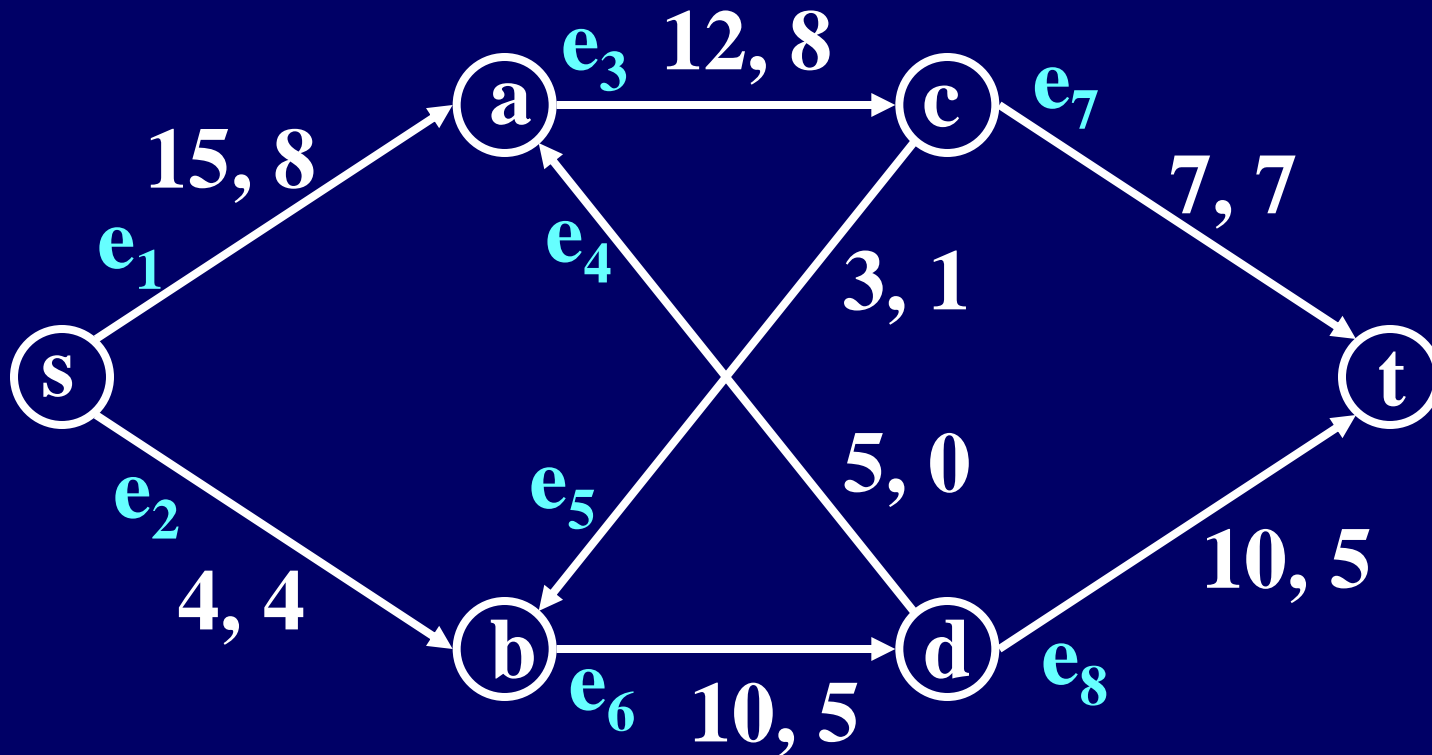
c_f

La chaîne augmentante



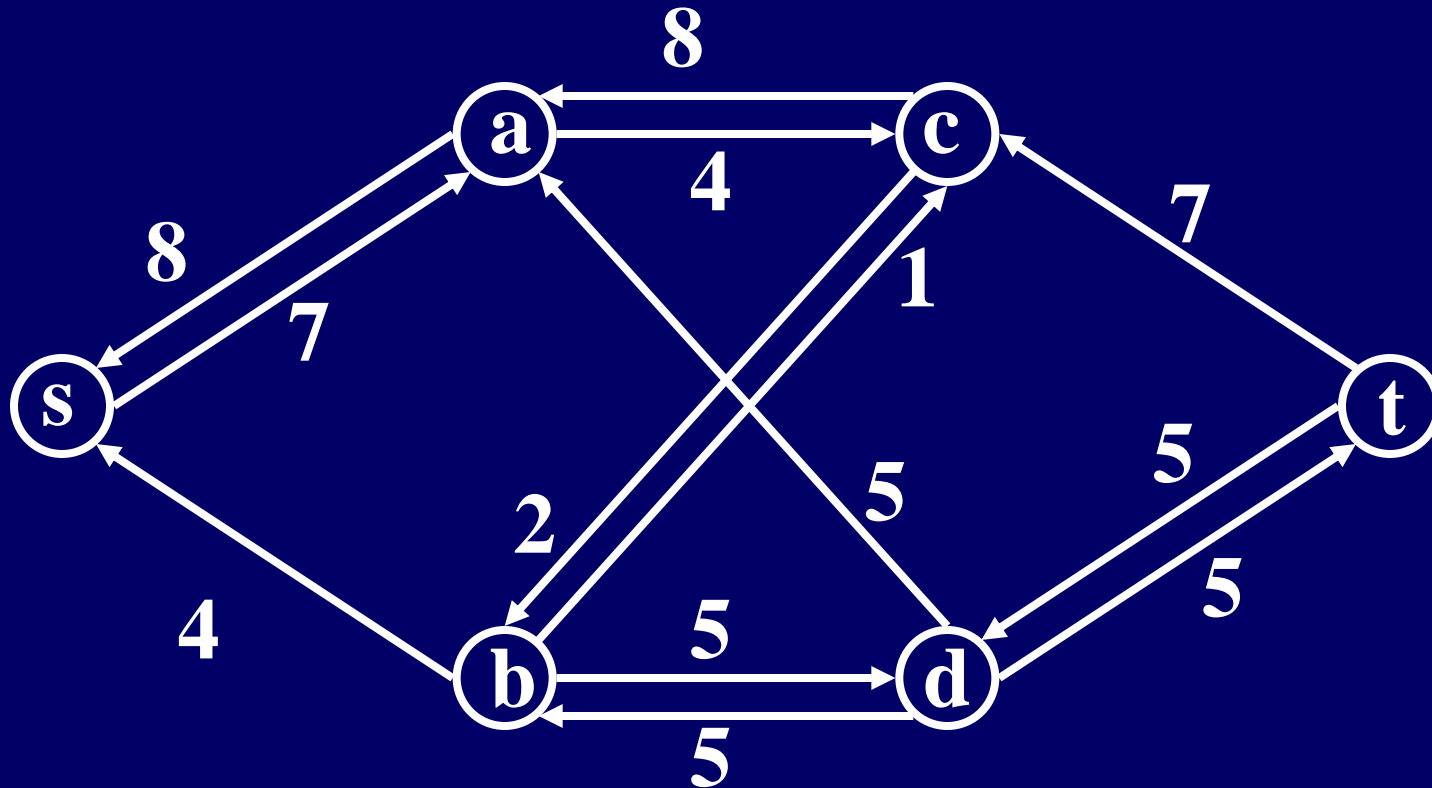
On peut faire une augmentation de $\min\{11,4,9\}=4$.

Le résultat



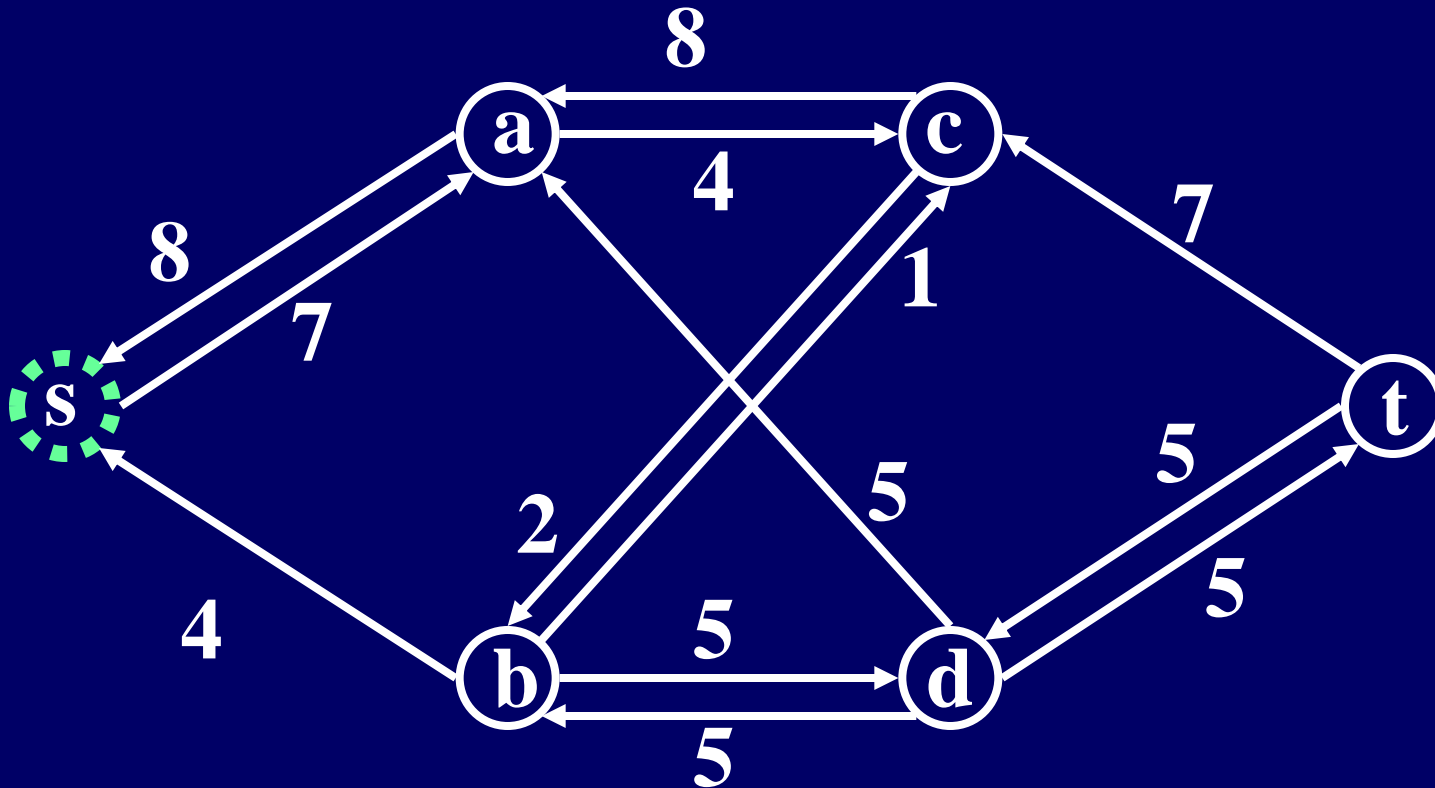
c, f

Le réseau résiduel



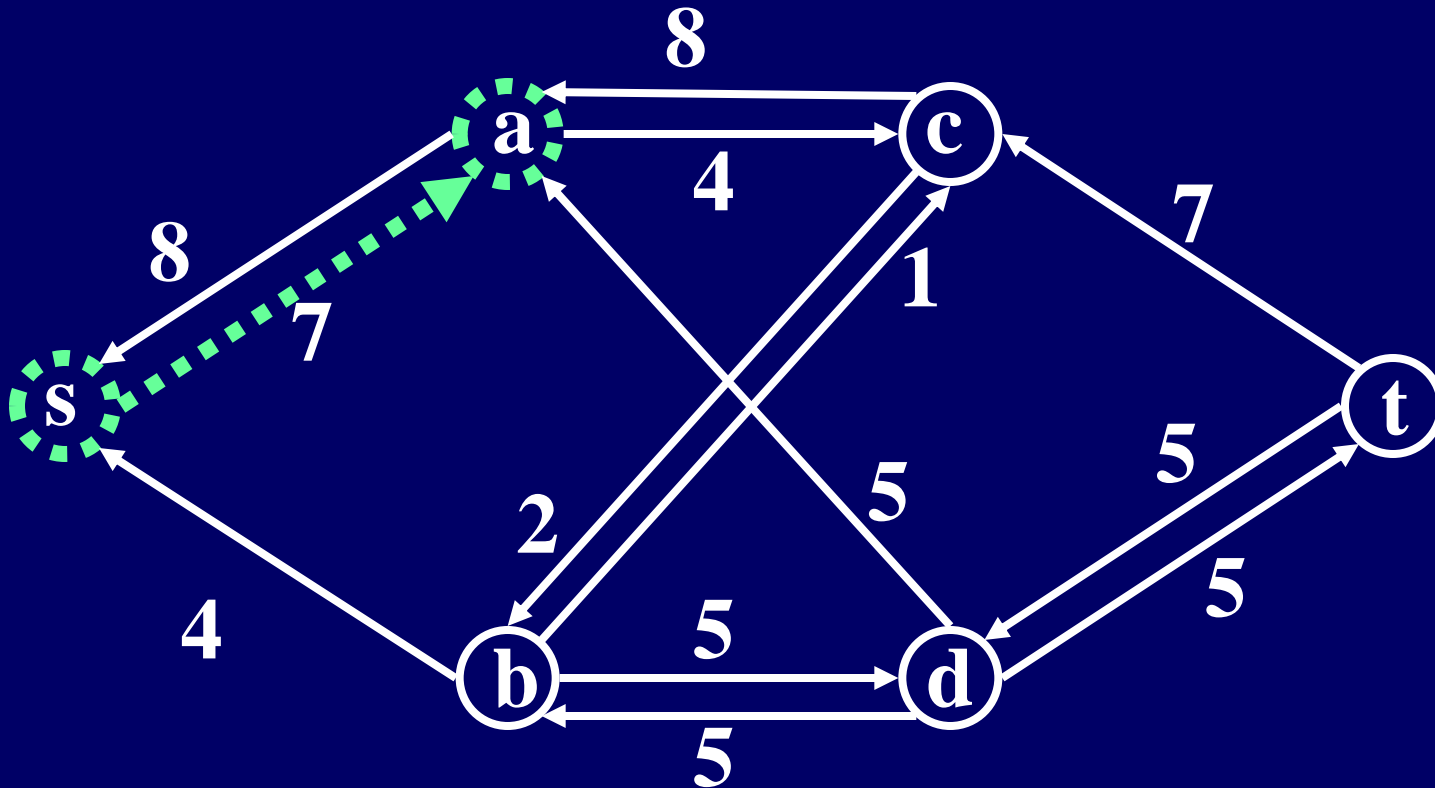
c_f

Recherche de chaîne augmentante



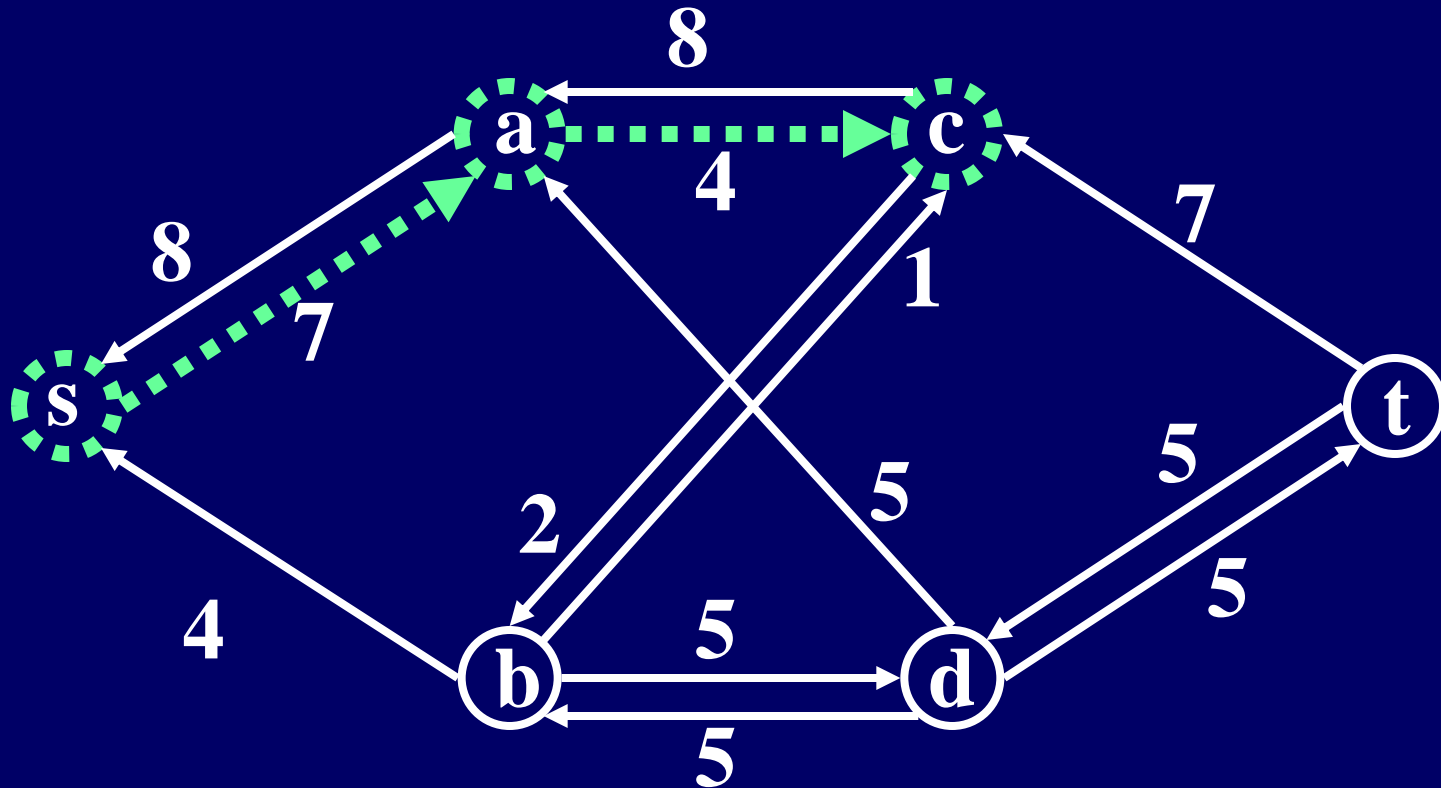
c_f

Recherche de chaîne augmentante



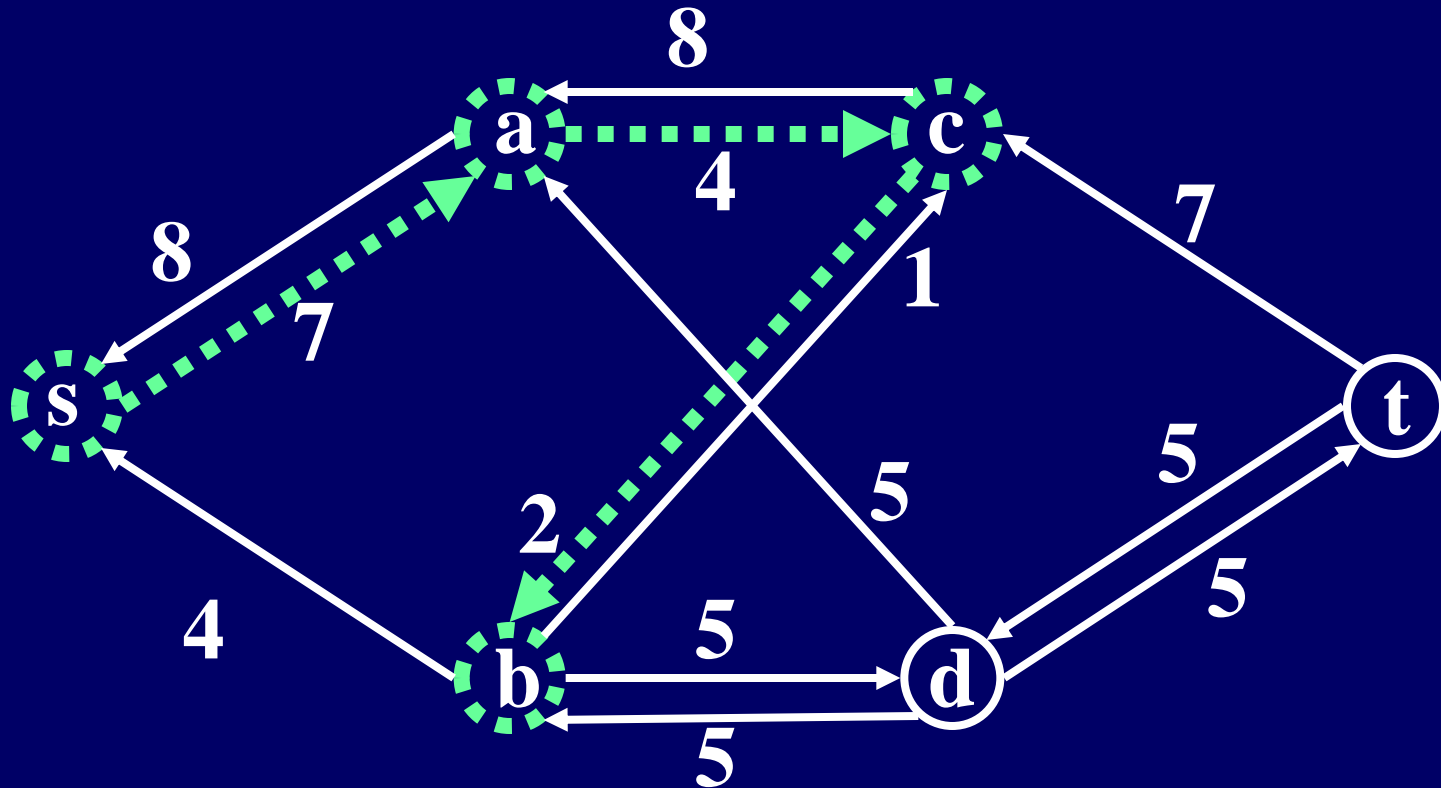
c_f

Recherche de chaîne augmentante



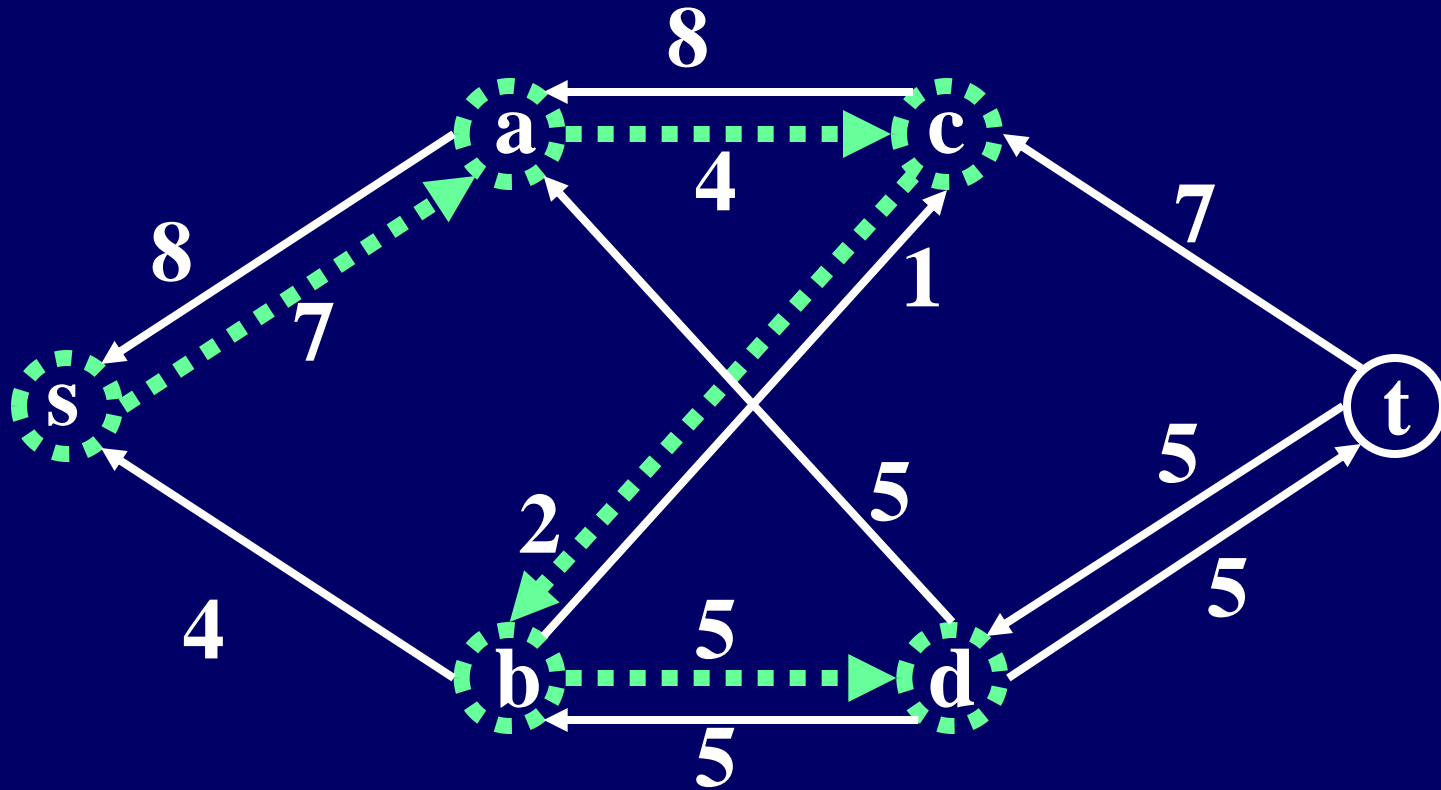
c_f

Recherche de chaîne augmentante



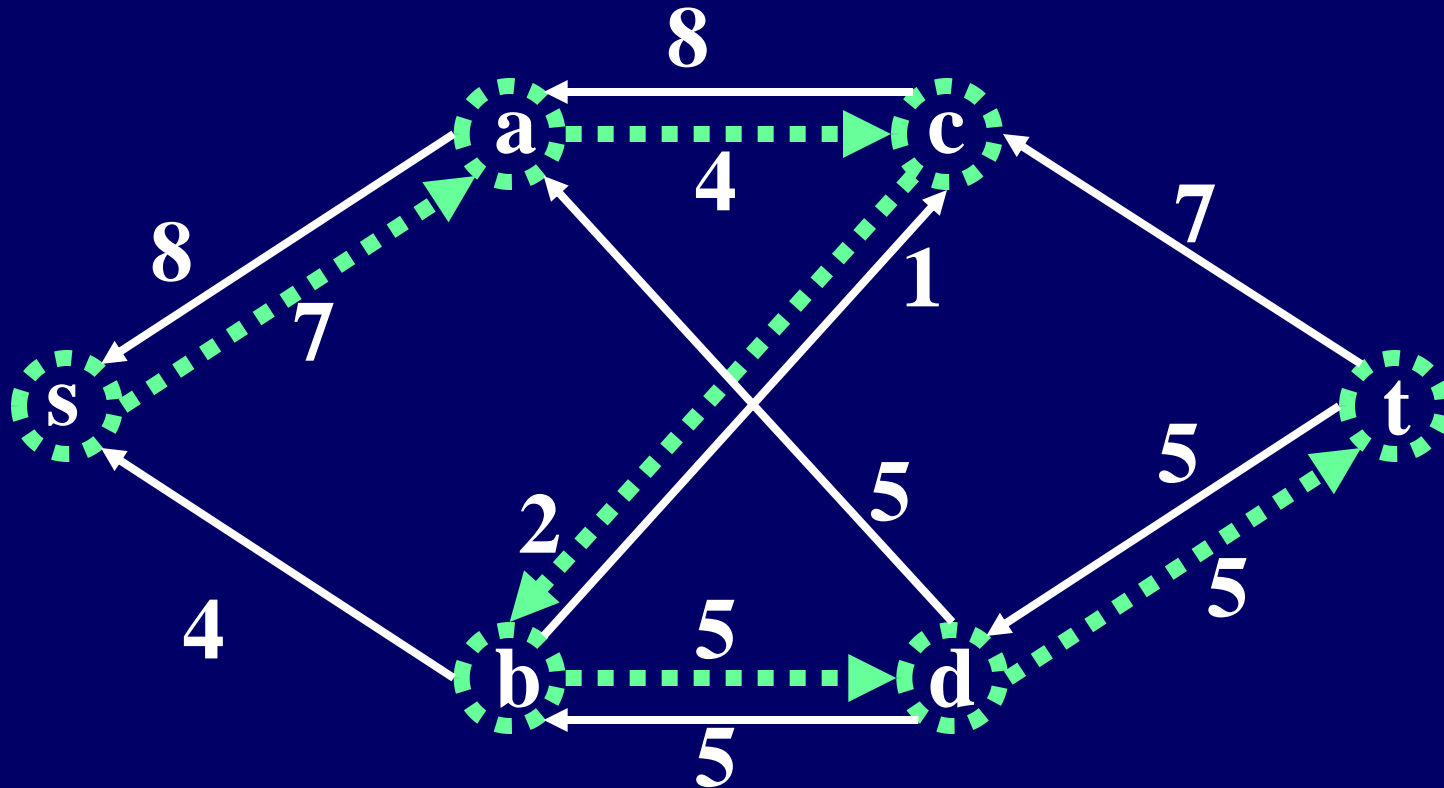
c_f

Recherche de chaîne augmentante



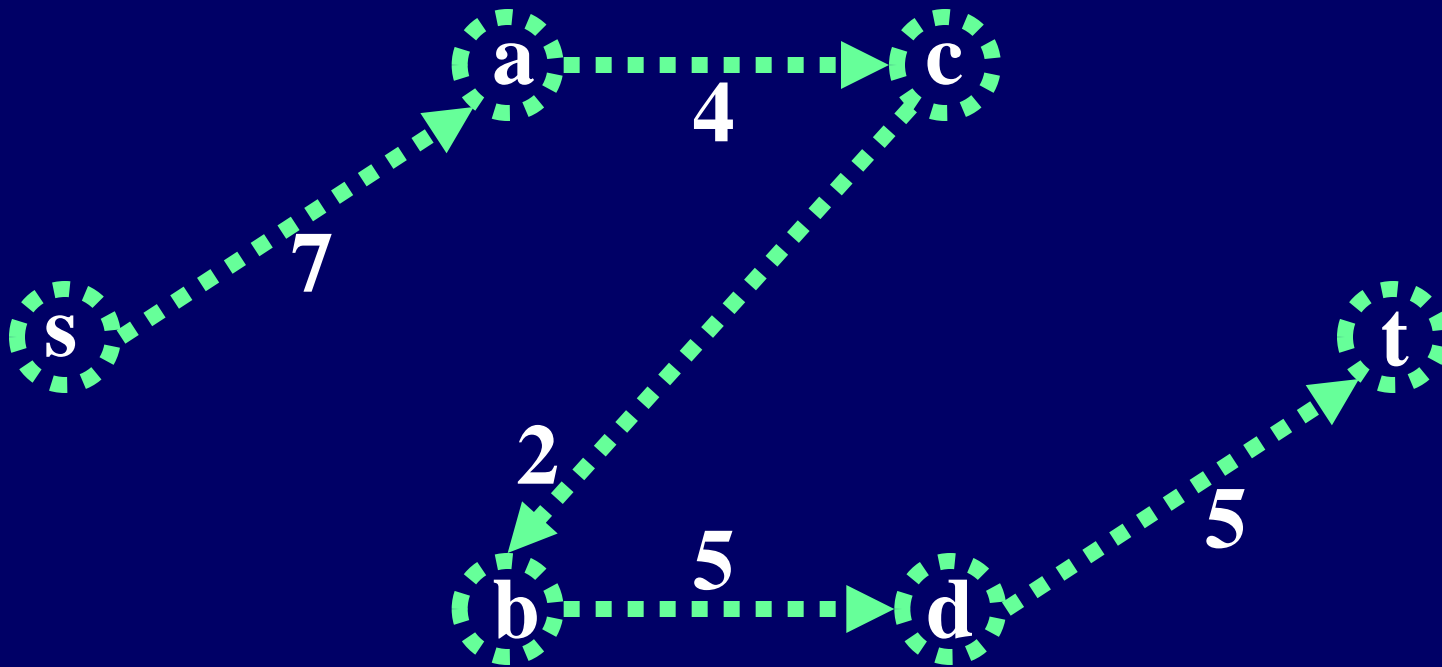
c_f

Recherche de chaîne augmentante



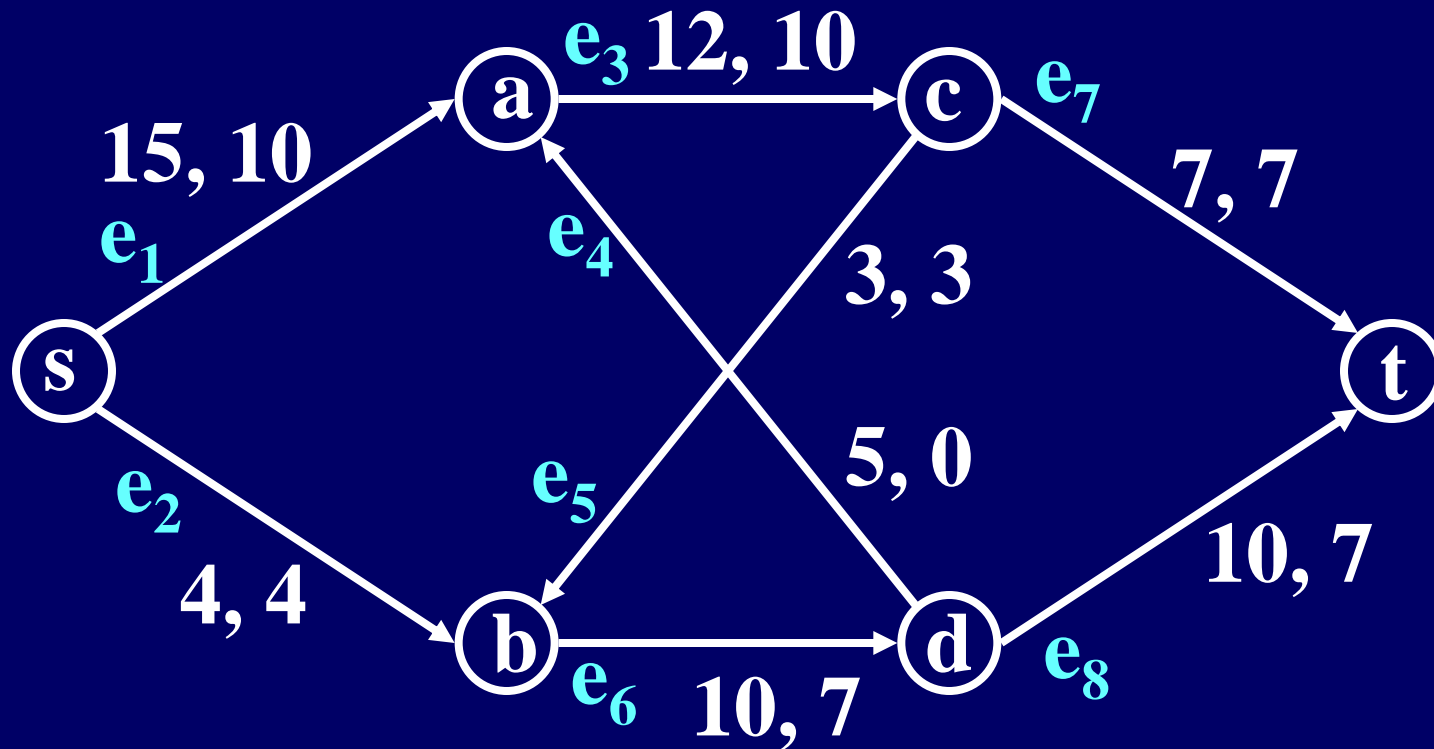
c_f

La chaîne augmentante



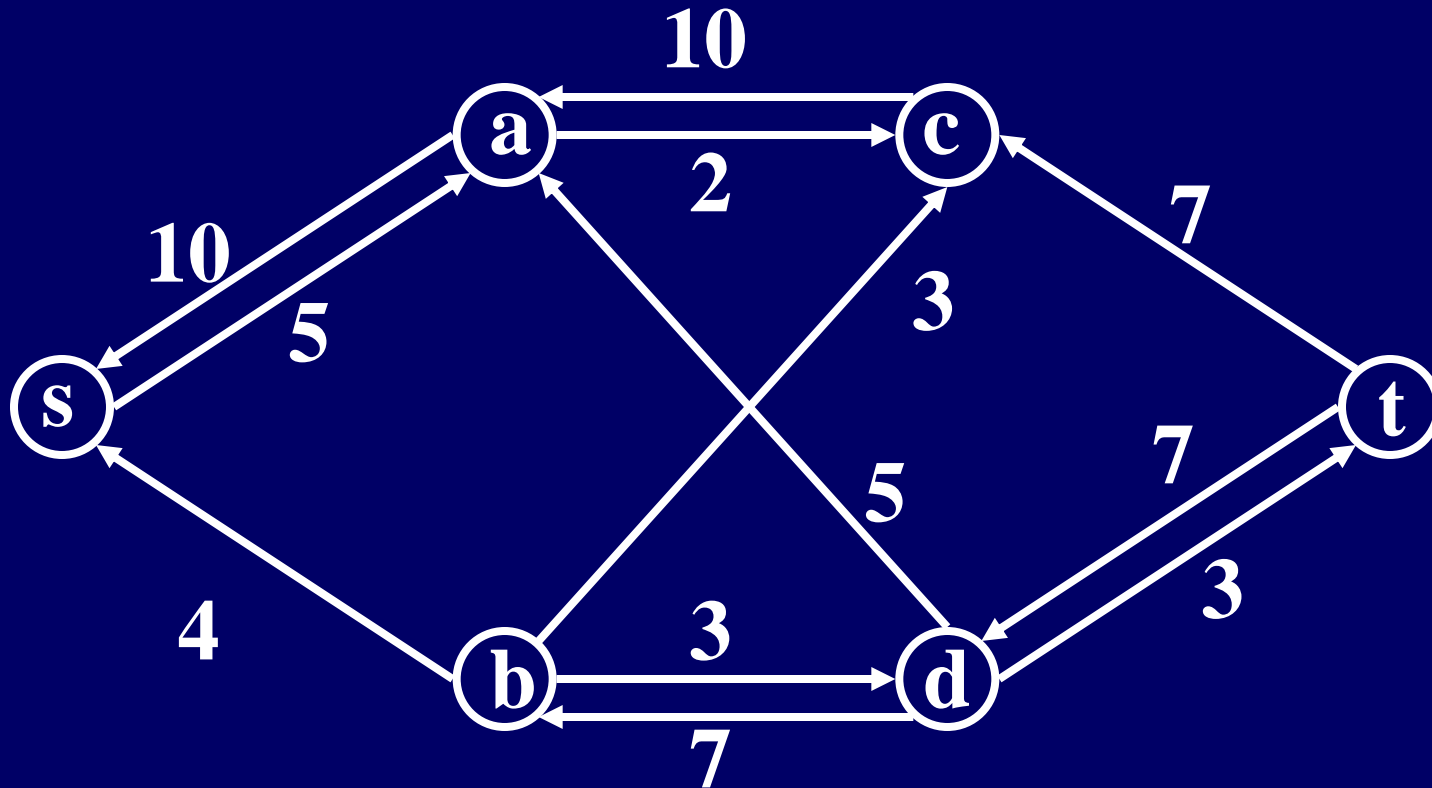
On peut faire une augmentation de $\min\{7,4,2,5,5\}=2$.

Le résultat



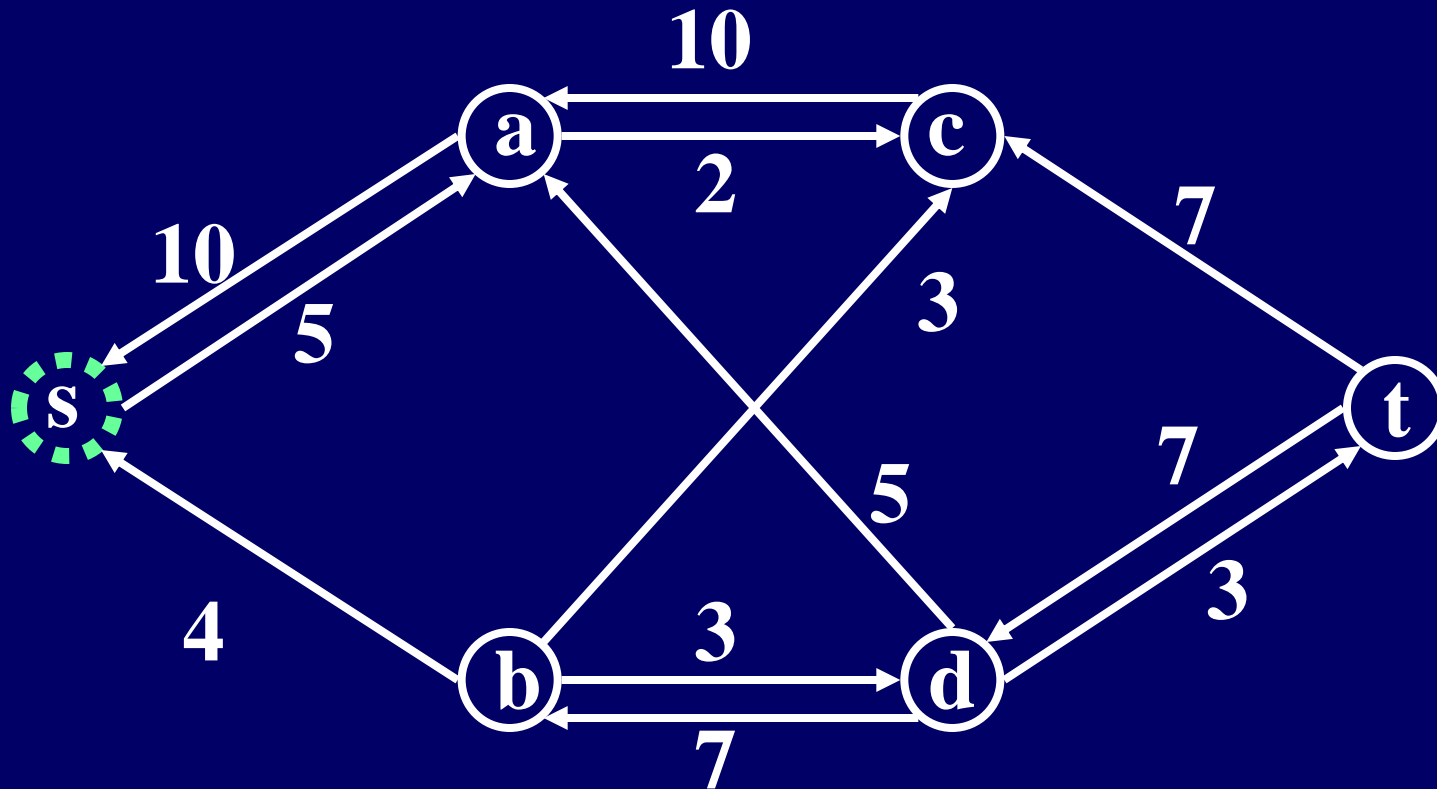
c, f

Le réseau résiduel



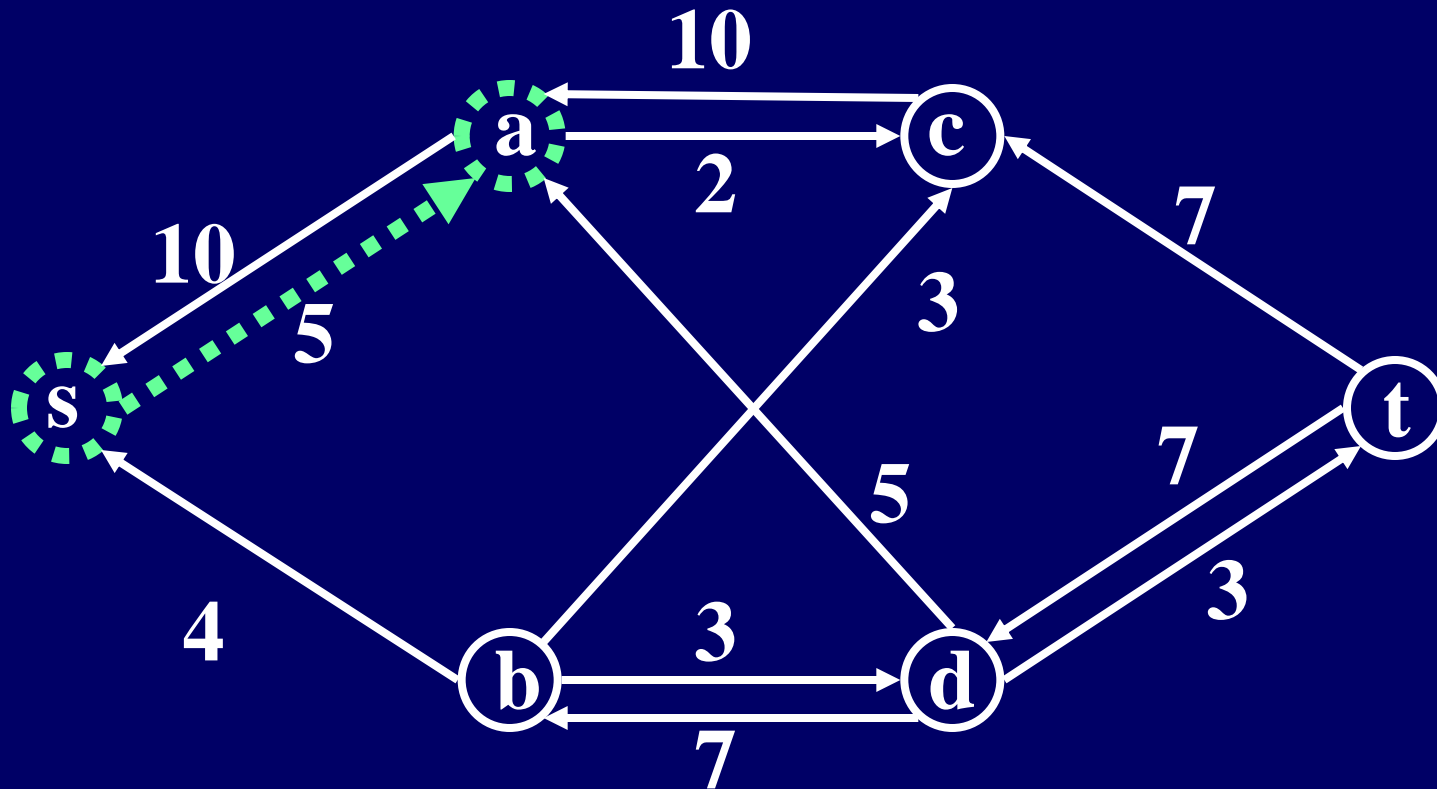
c_f

Recherche de chaîne augmentante



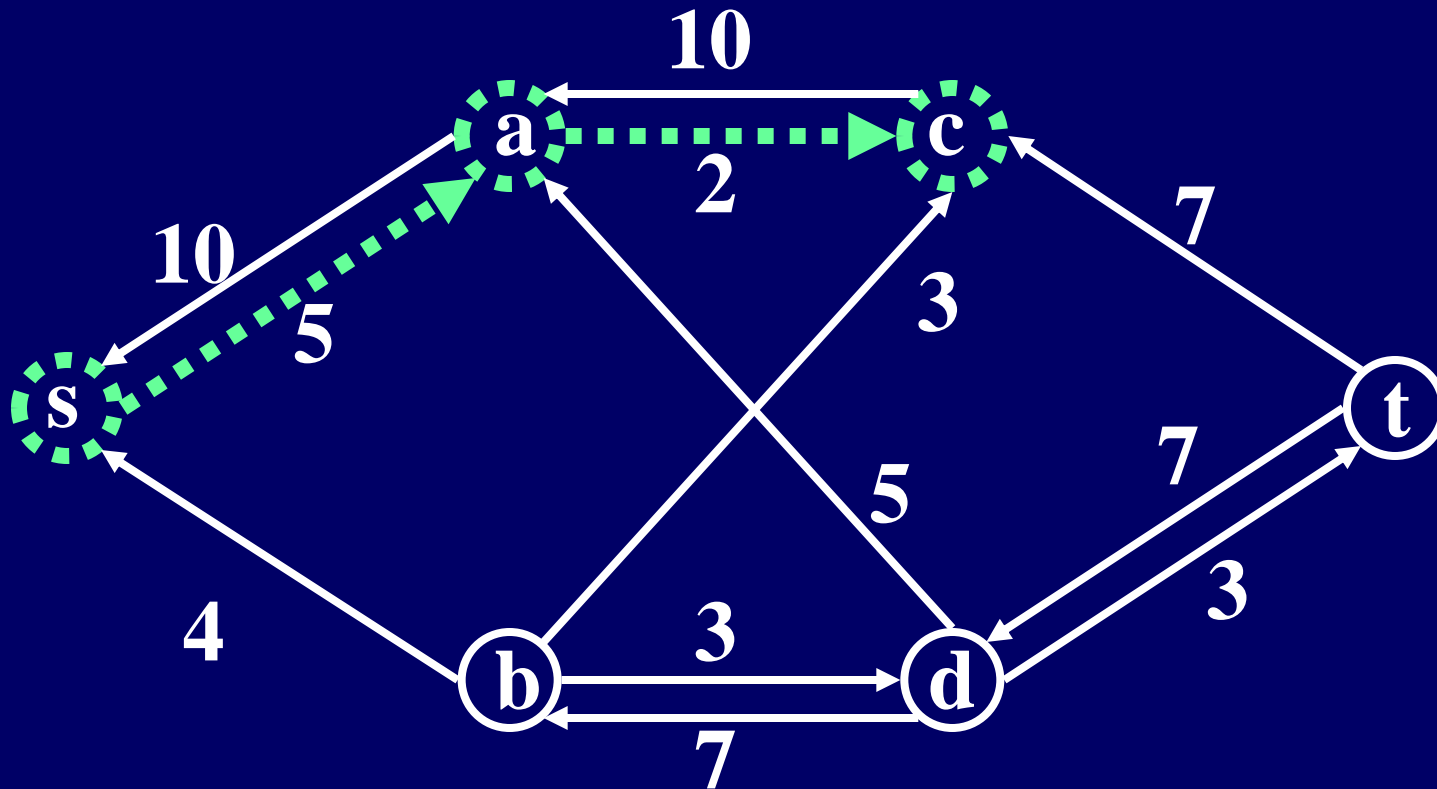
c_f

Recherche de chaîne augmentante



c_f

Recherche de chaîne augmentante



c_f

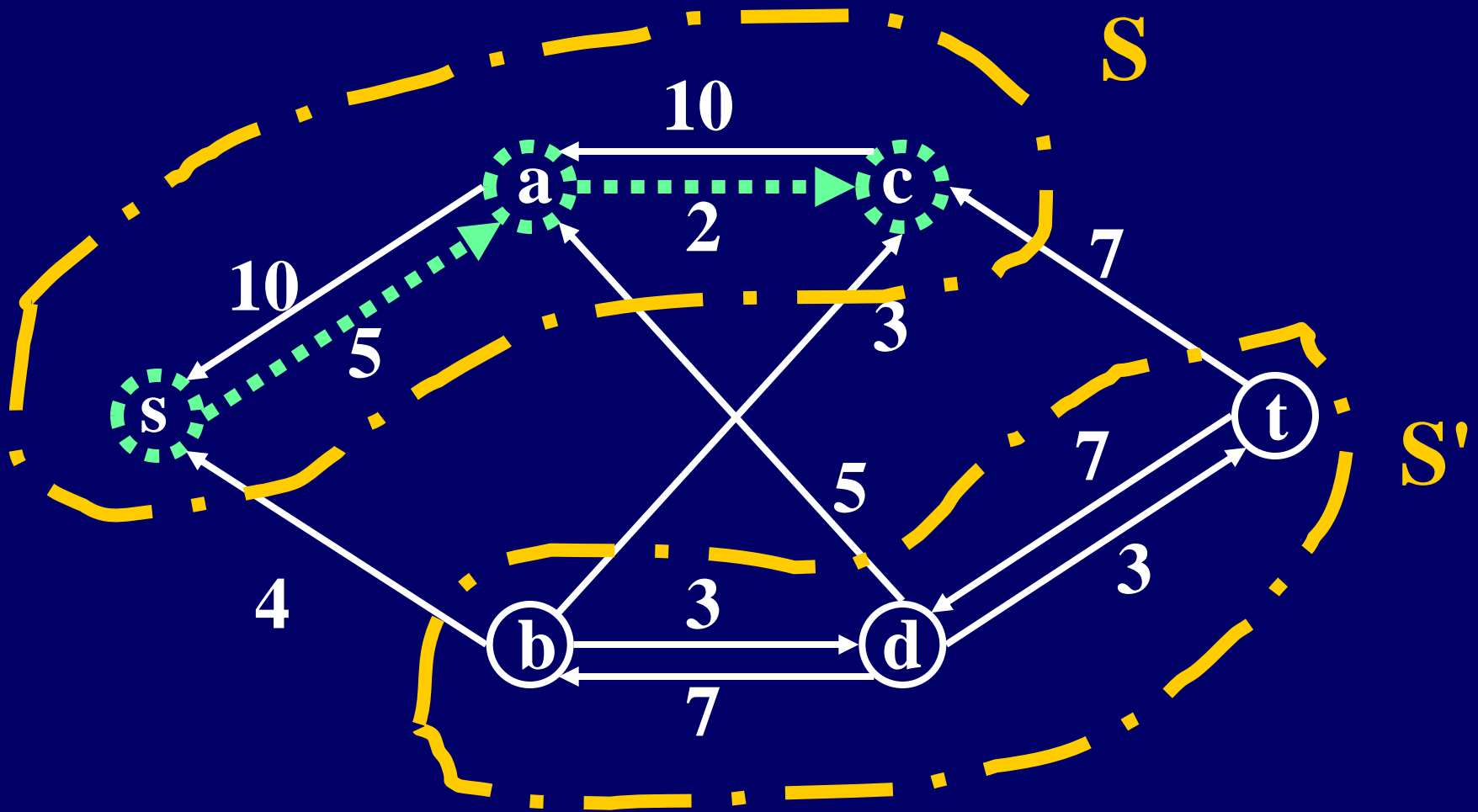
Recherche de chaîne augmentante

**Mais comment
continuer ?**

Recherche de chaîne augmentante

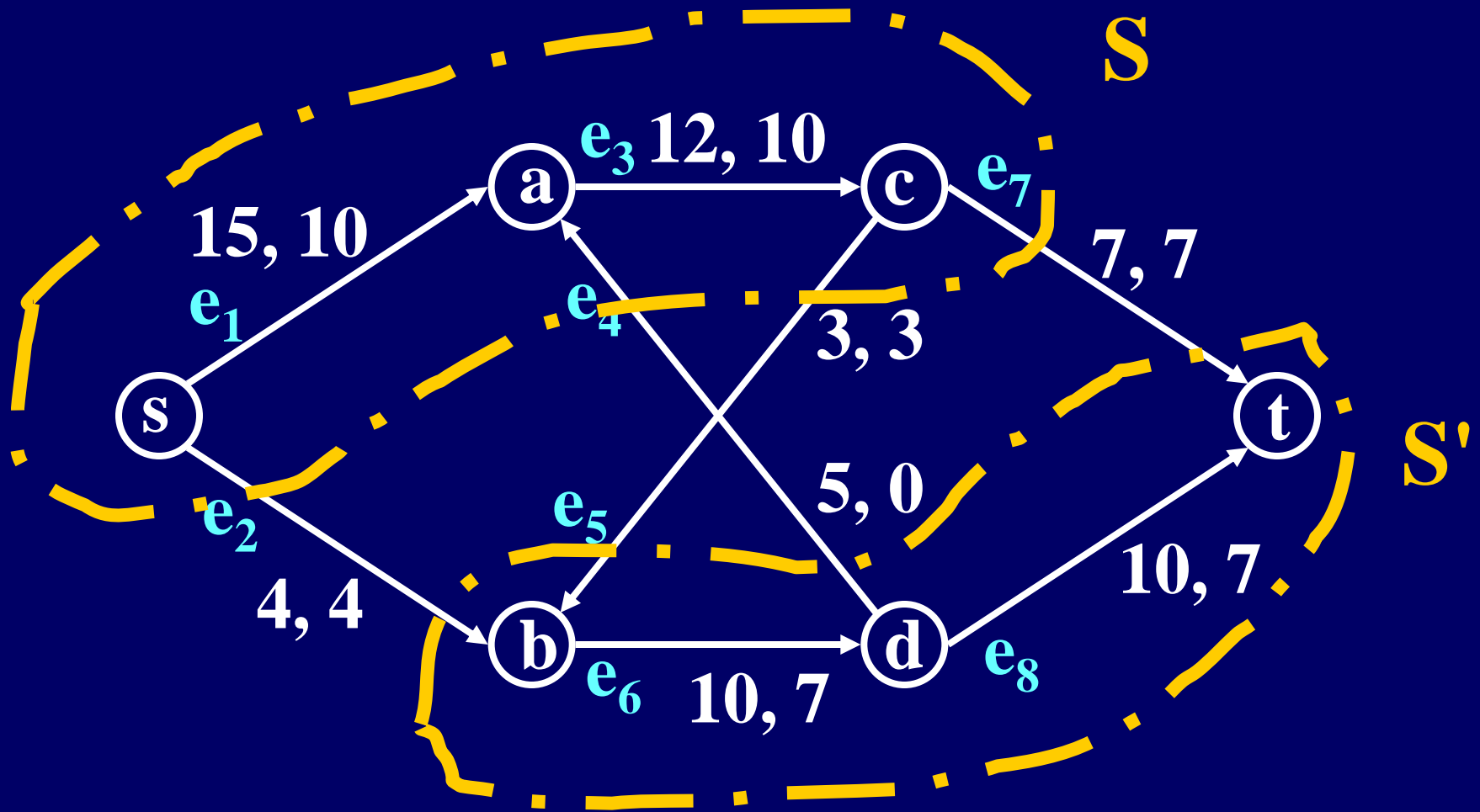
On ne peut pas !

Recherche de chaîne augmentante



Et toutes les arcs sont dans le sens $S' \rightarrow S$.

La coupe dans le résultat



Les arcs $S' \rightarrow S$ sont de flot nul et les arcs $S \rightarrow S'$ sont saturés.

Conclusion

Le flot est maximum !

L'algorithme est terminé !