Programmation dynamique

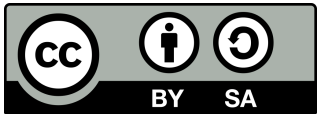
une histoire d'optimum

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

Un grand merci à D. Trystram et F. Wagner pour l'emprunt d'une partie des transparents

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

Algorithmique et Modélisation
L3-INFO



UGA Université Grenoble Alpes

PROGRAMMATION DYNAMIQUE

I. LE PROBLÈME : trouver une meilleure solution

II. UNE HISTOIRE : Al Khwarismi

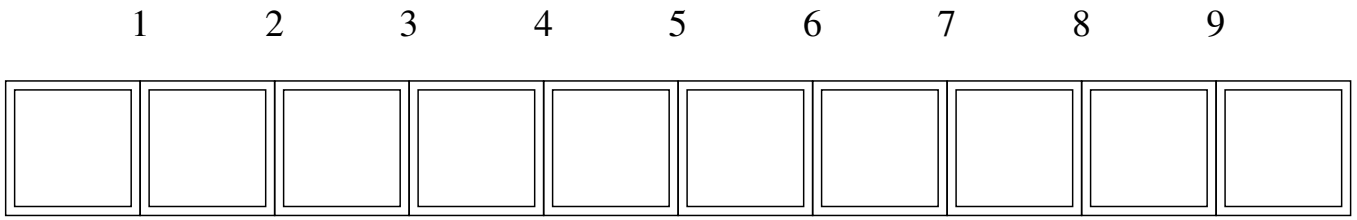
III. PRINCIPE : sous-optimalité, Richard Bellman

IV. SAC À DOS : un problème générique d'optimisation sous contrainte

V. SYNTHÈSE

UGA Université Grenoble Alpes

LA BARRE DE CHOCOLAT



Barre de longueur $L = 10$ positions des coupes $1, 2, \cdots, 9$

Objectif : Trouver une découpe de la barre de prix total maximal

taille de pièce : n prix
d'une pièce : $p(n)$

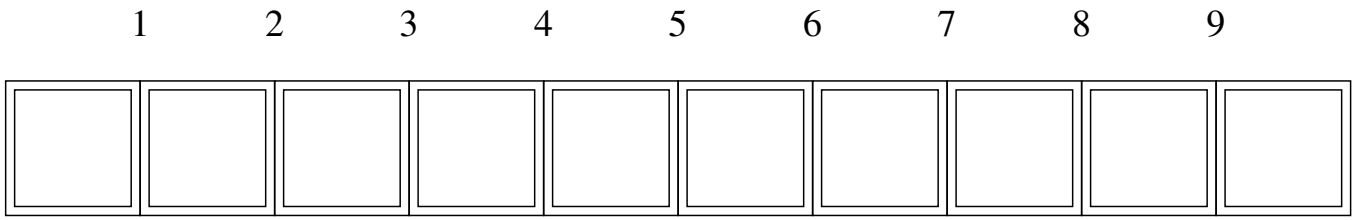
n	$p(n)$
0	0
1	2
2	3
3	8
4	10
5	13
6	15
7	16
8	21
⋮	⋮

Exercice

- Résoudre le problème pour $L = 8$

UGA Université Grenoble Alpes

LA BARRE DE CHOCOLAT



Barre de longueur $L = 10$ positions des coupes $1, 2, \cdots, 9$

Objectif : Trouver une découpe de la barre de prix total maximal

taille de pièce : n prix
d'une pièce : $p(n)$

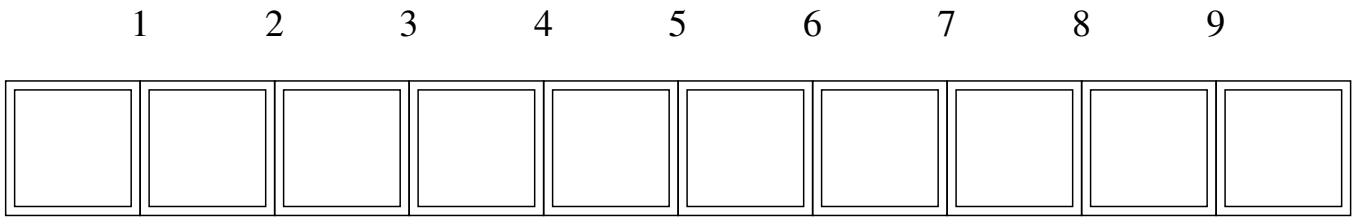
n	$p(n)$
0	0
1	2
2	3
3	8
4	10
5	13
6	15
7	16
8	21
⋮	⋮

Exercice

- Résoudre le problème pour $L = 8$
- Rechercher toutes les découpes possibles :

UGA Université Grenoble Alpes

LA BARRE DE CHOCOLAT



Barre de longueur $L = 10$ positions des coupes $1, 2, \cdots, 9$

Objectif : Trouver une découpe de la barre de prix total maximal

taille de pièce : n prix
d'une pièce : $p(n)$

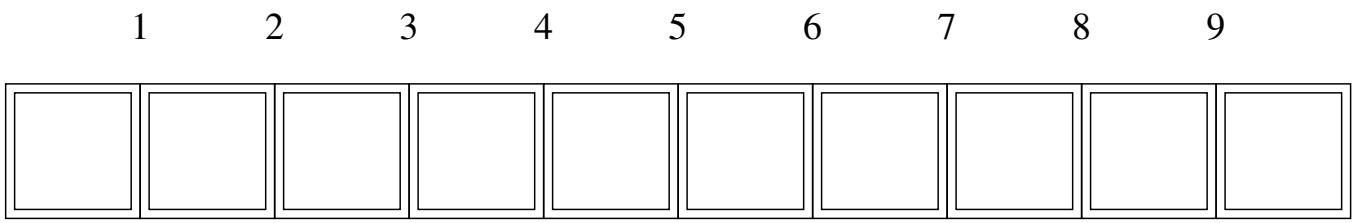
n	$p(n)$
0	0
1	2
2	3
3	8
4	10
5	13
6	15
7	16
8	21
⋮	⋮

Exercice

- Résoudre le problème pour $L = 8$
- Rechercher toutes les découpes possibles : 2^{n-1}

UGA Université Grenoble Alpes

FORMULATION DU PROBLÈME



Barre de longueur $L = 10$ positions des coupes $1, 2, \cdots, 9$

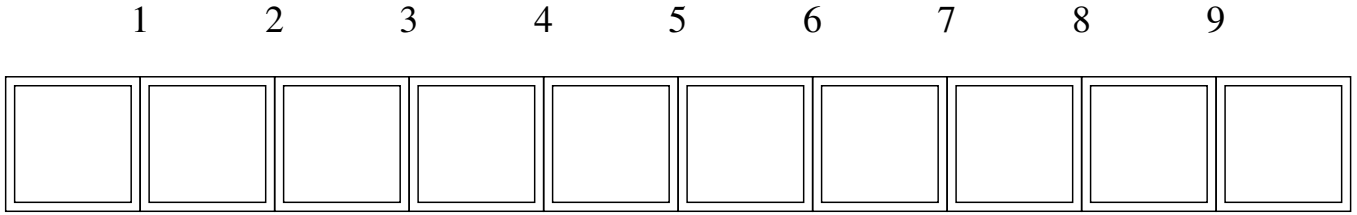
Une coupe en position k ramène le problème à 2 **sous-problèmes** trouver une découpe optimale d'une pièce de longueur k et une de longueur $L - k$
Notation $M(n)$ prix maximum d'une découpe d'une barre de longueur n.

Équation d'optimalité

$$\begin{aligned} M(0) &= 0 \\ M(n) &= \max\left\{p(n), \max_{1 \leqslant k \leqslant n-1} \{M(k) + M(n-k)\}\right\} \end{aligned}$$

Décomposition du problème en fonction de la position **d'une** position de découpe

AUTRE FORMULATION DU PROBLÈME



Barre de longueur $L = 10$ positions des coupes $1, 2, \cdots, 9$

Équation d'optimalité

$$\begin{aligned} M(0) &= 0 \\ M(n) &= \max\left\{p(n), \max_{1 \leqslant k \leqslant n-1} \{M(k) + M(n-k)\}\right\} \end{aligned}$$

Décomposition du problème en fonction de la position **d'une** position de découpe

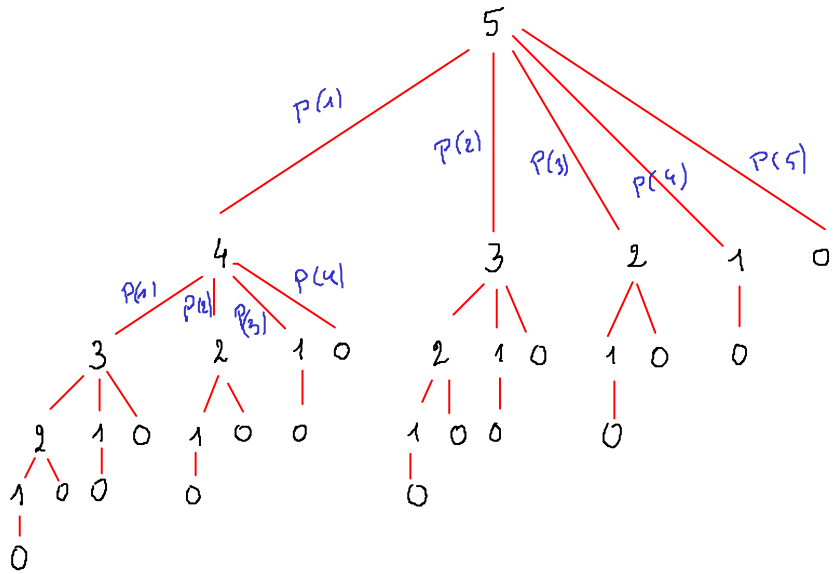
Autre équation d'optimalité

$$\begin{aligned} M(0) &= 0 \\ M(n) &= \max\left\{p(n), \max_{1 \leqslant k \leqslant n-1} \{p(k) + M(n-k)\}\right\} \end{aligned}$$

Décomposition du problème en fonction de la position **de la première** position de découpe.

UGA Université Grenoble Alpes

ARBRE DES APPELS POUR $L = 5$

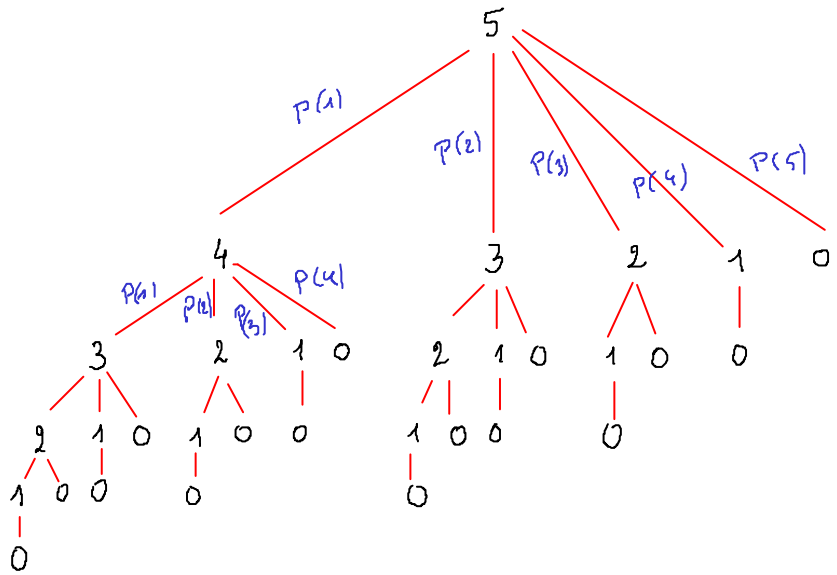


Arbre des appels pour $L = 5$

Le prix associé à une branche (feuille) est la somme des prix associés à chaque décision (arêtes)
Coût de l'algorithme :

$$C_{rec}(n) = \sum_{k=1}^{n-1} C_{rec}(i) + 1$$

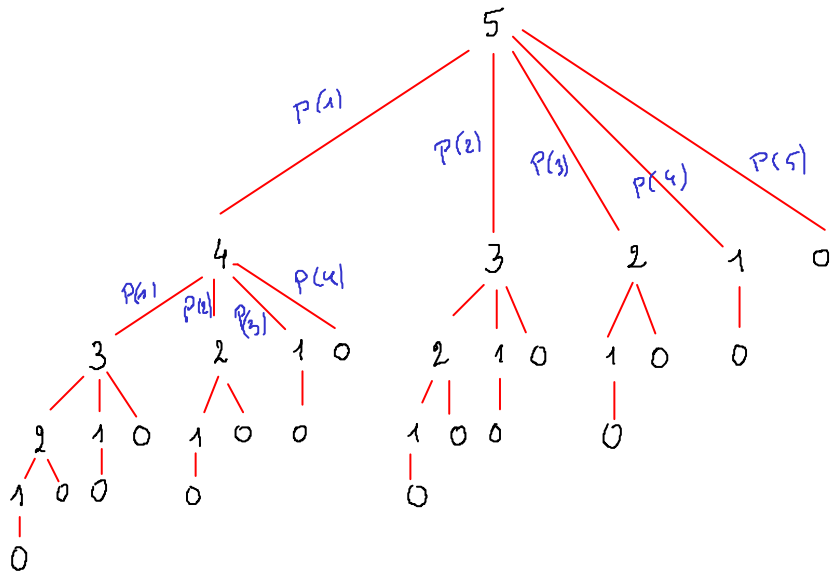
REDONDANCE ET MÉMOÏSATION



Arbre des appels pour $L = 5$

Valeurs à mémoïser : association paramètre d'appel et valeur de retour
Ici 5 valeurs à garder

REDONDANCE ET MÉMOÏSATION



Arbre des appels pour $L = 5$

Valeurs à mémoïser : association paramètre d'appel et valeur de retour
Ici 5 valeurs à garder
Coût de l'algorithme

$$C_{rec-mem}(n) = 1 + 2 + \cdots + n - 1 = \frac{n(n-1)}{2}$$ et en mémoire ?

ÉTENDRE LE PROBLÈME

Passage à 2 paramètres :

- L taille de la barre
- n taille maximale des pièces découpées

$M(L, n)$ prix de la découpe optimale

$$M(L,n) = \begin{cases} \max\{p(n) + M(L-n, n), M(L, n-1)\} & \text{si } L \geqslant n \\ M(L, n-1) & \text{si } L < n \end{cases}$$

$$M(0,0) = 0$$

Difficulté : comportement au bord

CONSTRUCTION DE LA SOLUTION

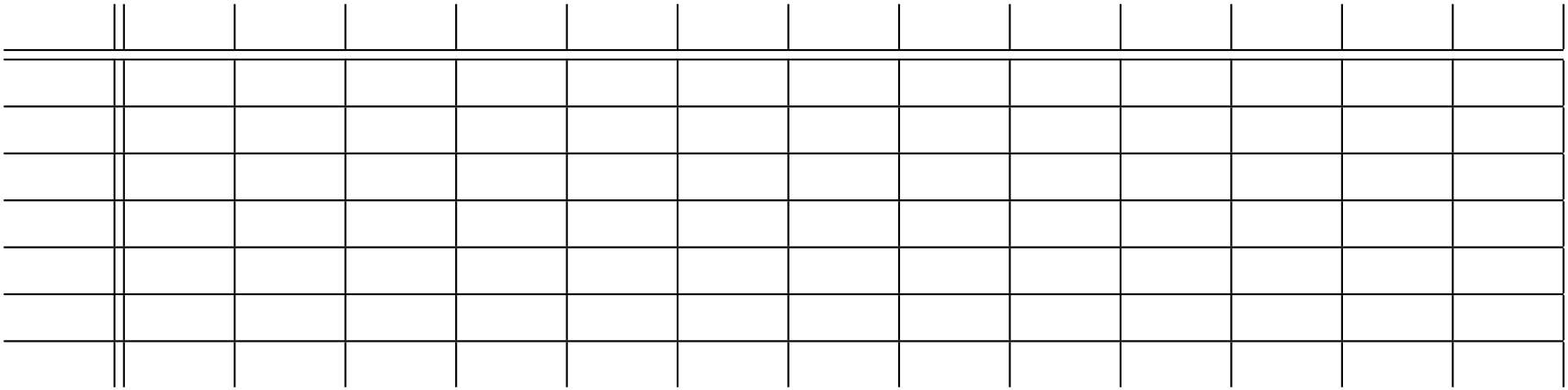
Taille	*Prix*	0	1	2	3	4	5	6	7	8	9	10	···
1	2	0	2	4	6	8	10	12	14	16	18	20	···
2	3												
3	8												
4	10												
5	13												
6	15												
7	16												
8	21												
⋮	⋮												

CONSTRUCTION DE LA SOLUTION

Taille	*Prix*	0	1	2	3	4	5	6	7	8	9	10	···
1	2	0	2	4	6	8	10	12	14	16	18	20	···
2	3	0	2	4	6	8	10	12	14	16	18	20	···
3	8	0	2	4	8	10	12	16	18	20	24	26	···
4	10	0	2	4	8	10	12	16	18	20	24	26	···
5	13	0	2	4	8	10	13	16	18	21	24	26	···
6	15	0	2	4	8	10	13	16	18	21	24	26	···
7	16	0	2	4	8	10	13	16	18	21	24	26	···
8	21	0	2	4	8	10	13	16	18	21	24	26	···
⋮	⋮												

UGA Université Grenoble Alpes

MÉTHODE DE CALCUL



Balayage

- par ligne
- par colonne
- en diagonale
- ...

PROGRAMMATION DYNAMIQUE

I. LE PROBLÈME : trouver une meilleure solution

II. UNE HISTOIRE : Al Khwarismi

III. PRINCIPE : sous-optimalité, Richard Bellman

IV. SAC À DOS : un problème générique d'optimisation sous contrainte

V. SYNTHÈSE

BAGDAD, IX SIÈCLE

On rapporte l'histoire d'un chamelier qui négocia le prix d'une caravane avec un émissaire du calife El Maamun, fils du célèbre Harun al Rashid [1].
L'affaire fut conclue, la formule inscrite sur le sable : $(1 + 2) \times 3 \times 4 + 5 = 41$ barils d'huile.
Le temps d'une prière, le vent efface les parenthèses : $1 + 2 \times 3 \times 4 + 5$.
Arrivé au palais, il a fallu recalculer la valeur. Le chamelier proposa (évidemment) $(1 + 2) \times 3 \times (4 + 5) = 81$.

1. Giegerich et al. Science of Computer Programming 2004

RÉSOLUTION "À LA MAIN"

Al Khwarizmi fut appelé au palais pour tirer au clair cette affaire.
Il trouva tout d'abord qu'il y avait 14 façons différentes de parenthéser l'expression.
$(1 + 2) \times 3 \times (4 + 5) = 81$
$1 + 2 \times (3 \times 4 + 5) = 35$
$(1 + 2 \times 3) \times 4 + 5 = 33$
...
Bien sûr, il est possible de calculer toutes les valeurs et d'en prendre le minimum (valeur qui avantageait le Calife), mais Al Khwarizmi proposa une méthode générale [a].
Pour cela, il reçu une bourse d'étude [b].

a. Il n'est pas le "père de l'algorithmique" sans raison !
b. Il n'y avait pas encore l'ANR à cette époque...

FORMALISATION DU PROBLÈME

On note τ l'expression complète (la chaine de caractères) : $\tau = 1 + 2 \times 3 \times 4 + 5$ et on indice les caractères $\tau =_0 1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$.
Ainsi, $t(i, j)$ désigne la sous-expression entre i et j. Par exemple, $t(2, 5) = 2 \times 3$.
La *meilleure* valeur (minimale pour le Calife) de la sous-expression est stockée dans un tableau (T). L'objectif est de déterminer $T(0, 9)$.

RÉSOLUTION

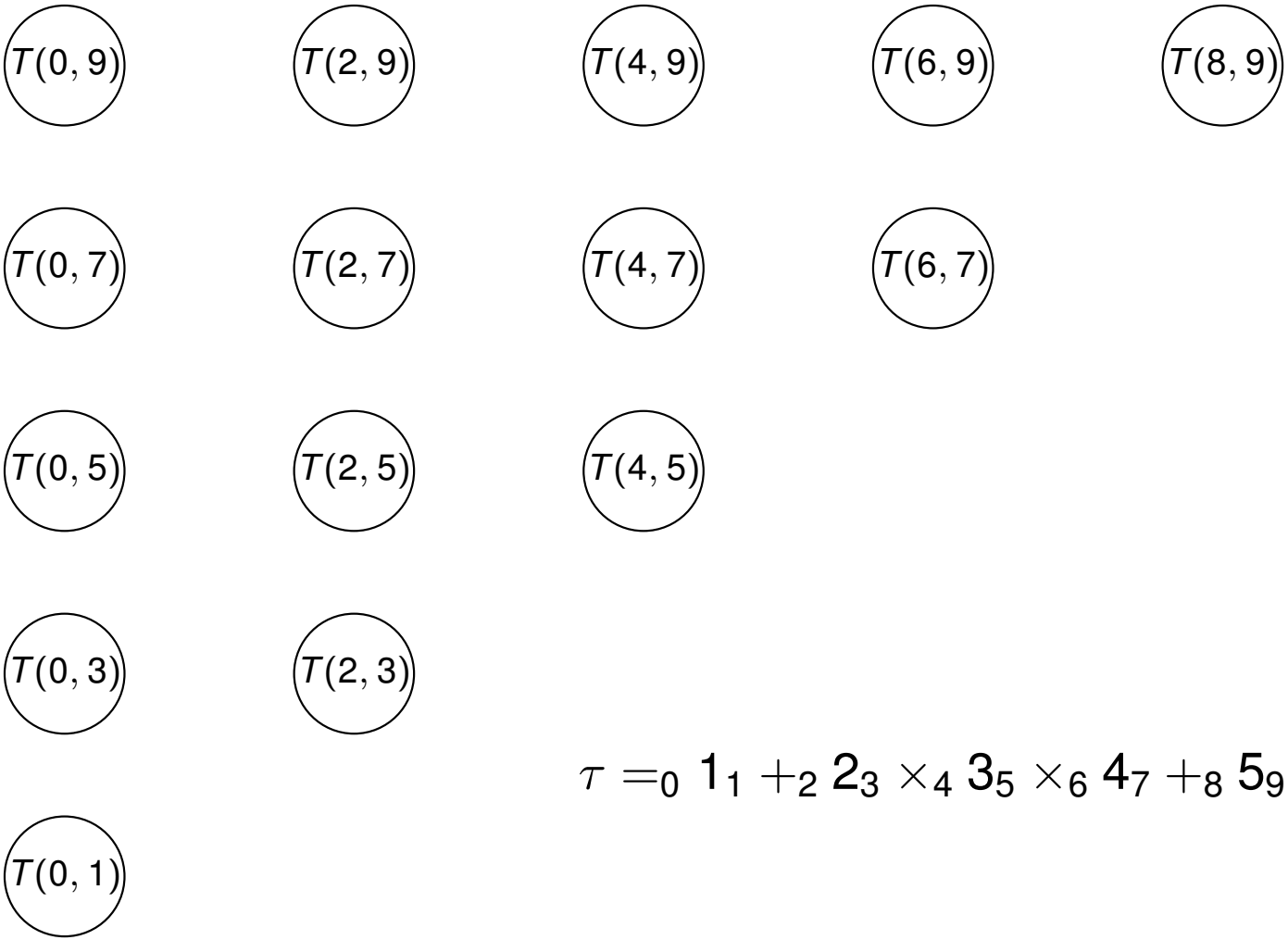
On définit

- $T(i, i+1) = n$ si $t(i, i+1) = n$
- $T(i, j) = min_{k|i<k<j}[T(i, k)op(k, k+1)T(k+1, j)]$

Ainsi, partant des petites valeurs de $((i, i+1)$, on peut calculer successivement toutes les entrées de la table T.
$T(0, 9) = min[T(0, 1) + T(2, 9), T(0, 3) \times T(4, 9), T(0, 5) \times T(6, 9), T(0, 7) + T(8, 9)]$.

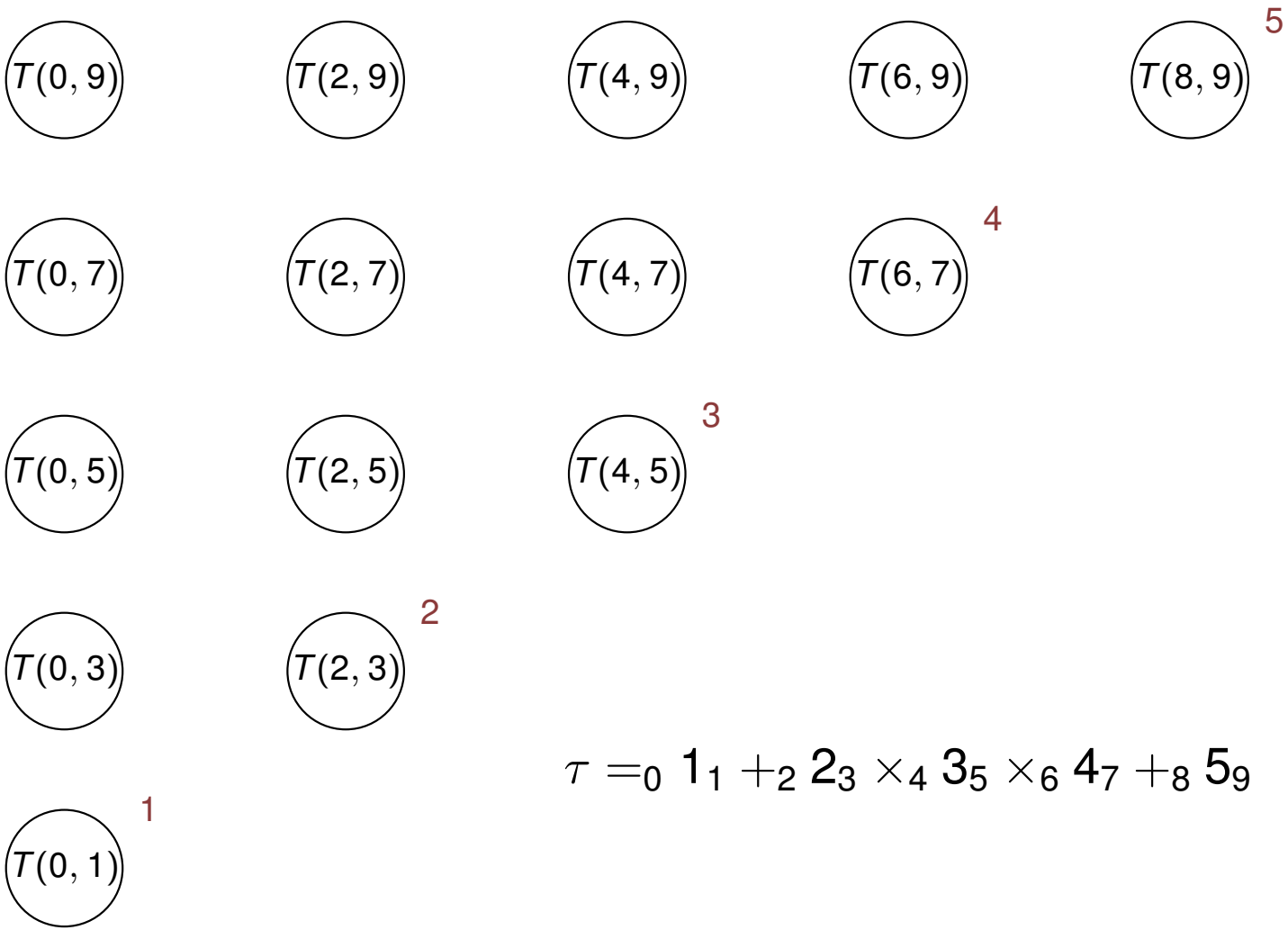
UGA Université Grenoble Alpes

DÉROULEMENT



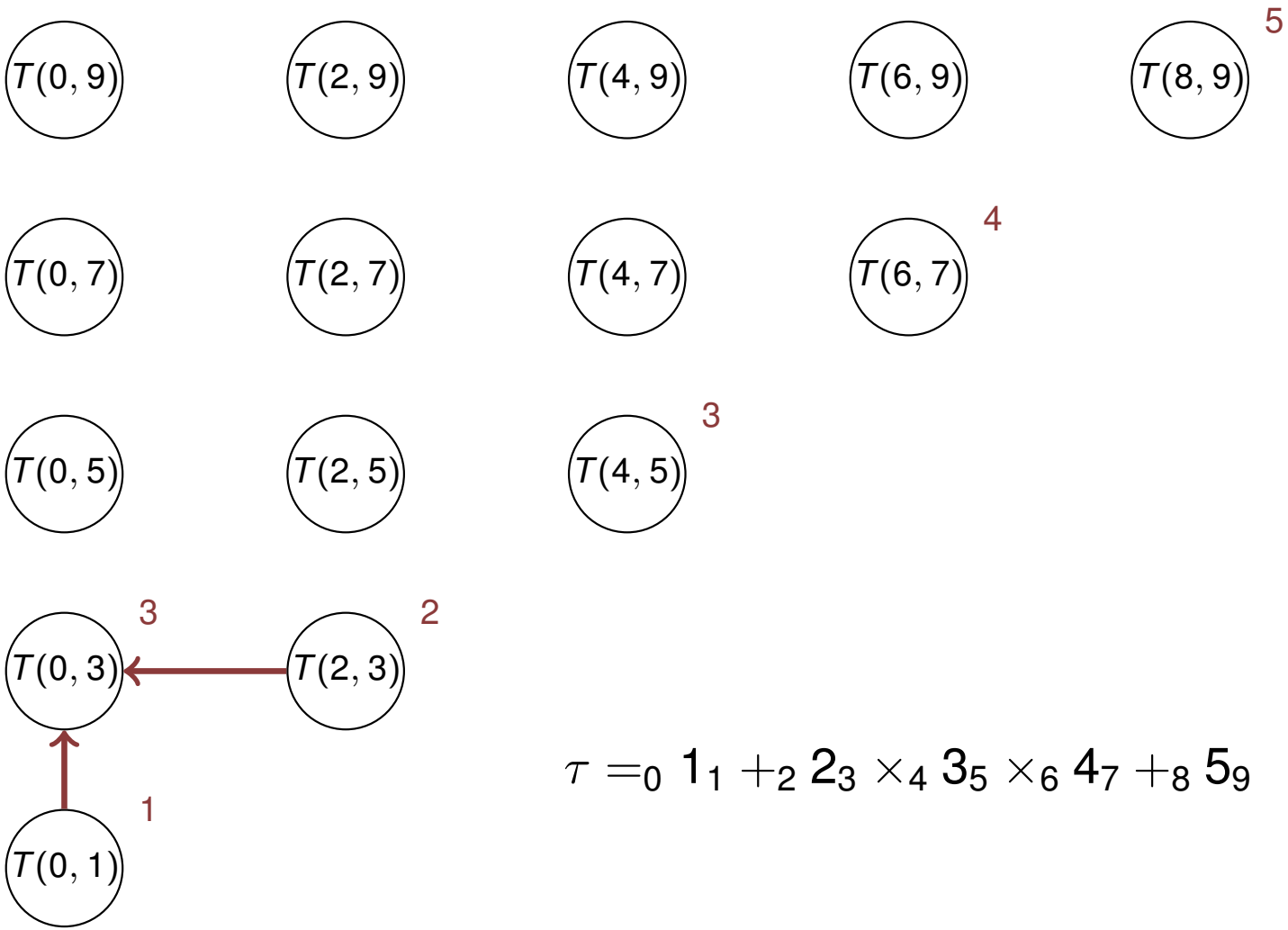
$$\tau =_0 1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



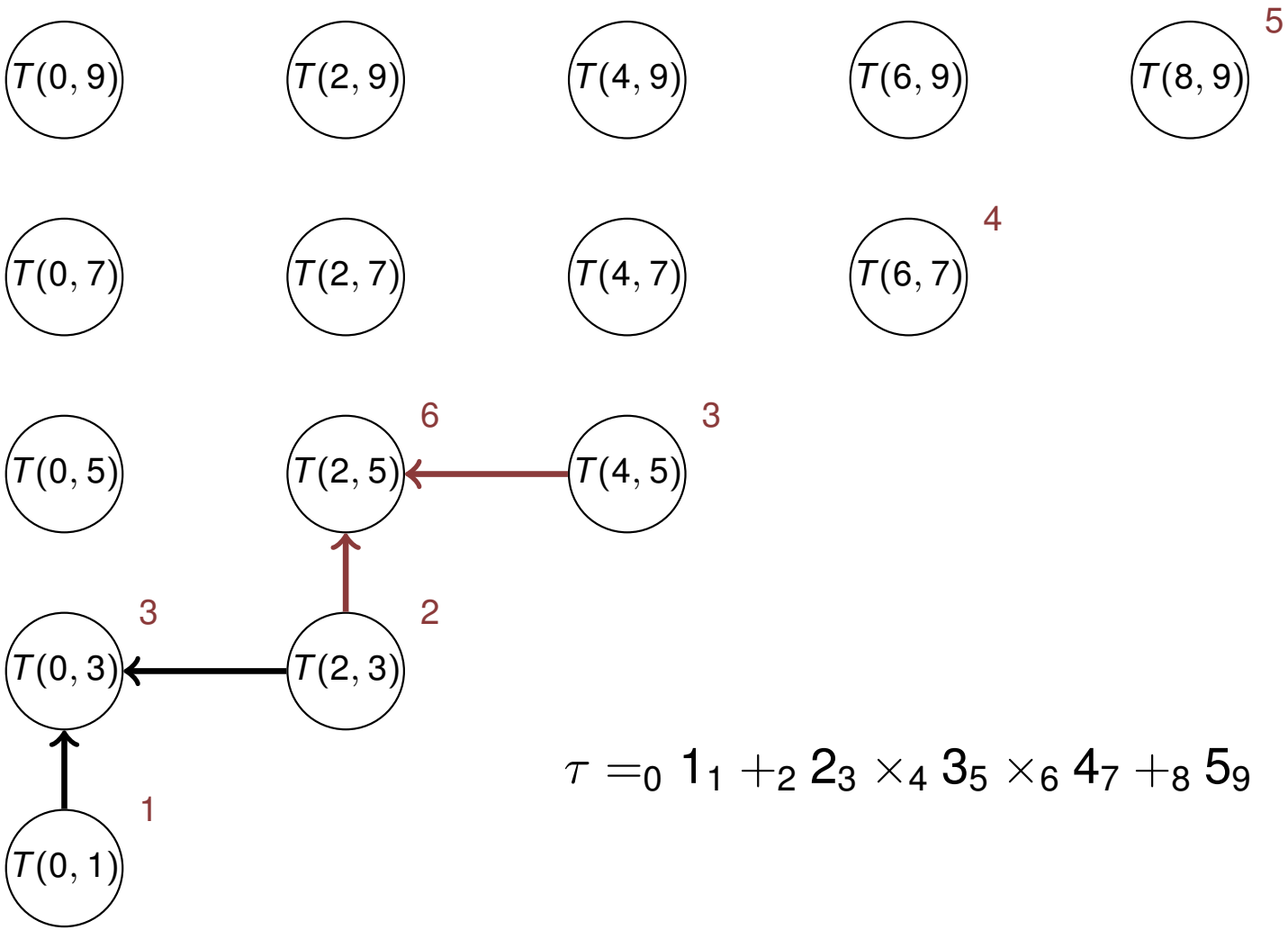
$$\tau = {}_0\,1_1 \,+_2\, 2_3 \,\times_4\, 3_5 \,\times_6\, 4_7 \,+_8\, 5_9$$

DÉROULEMENT



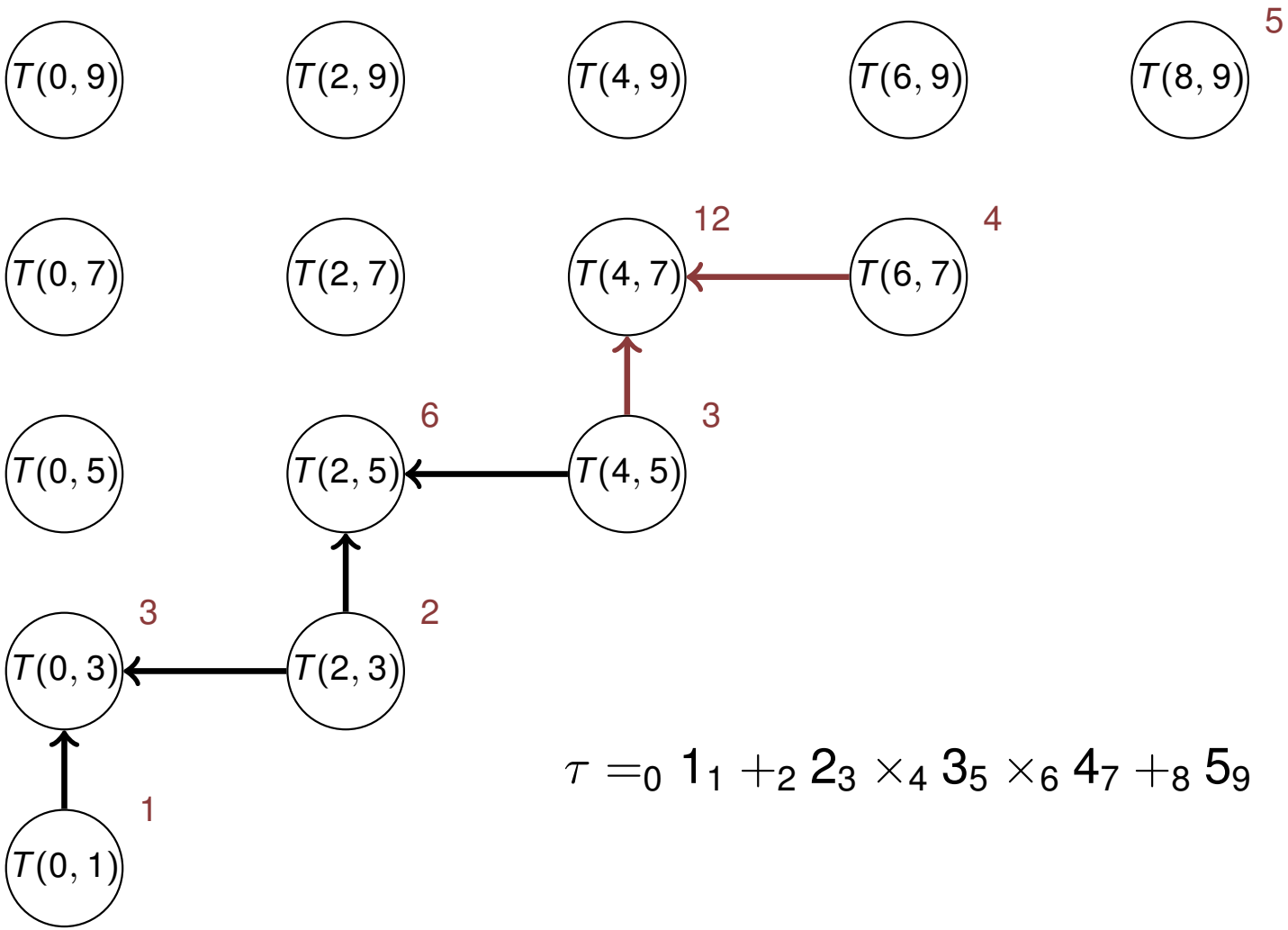
$$\tau = 0\ 1_1 + 2\ 2_3 \times 4\ 3_5 \times 6\ 4_7 + 8\ 5_9$$

DÉROULEMENT



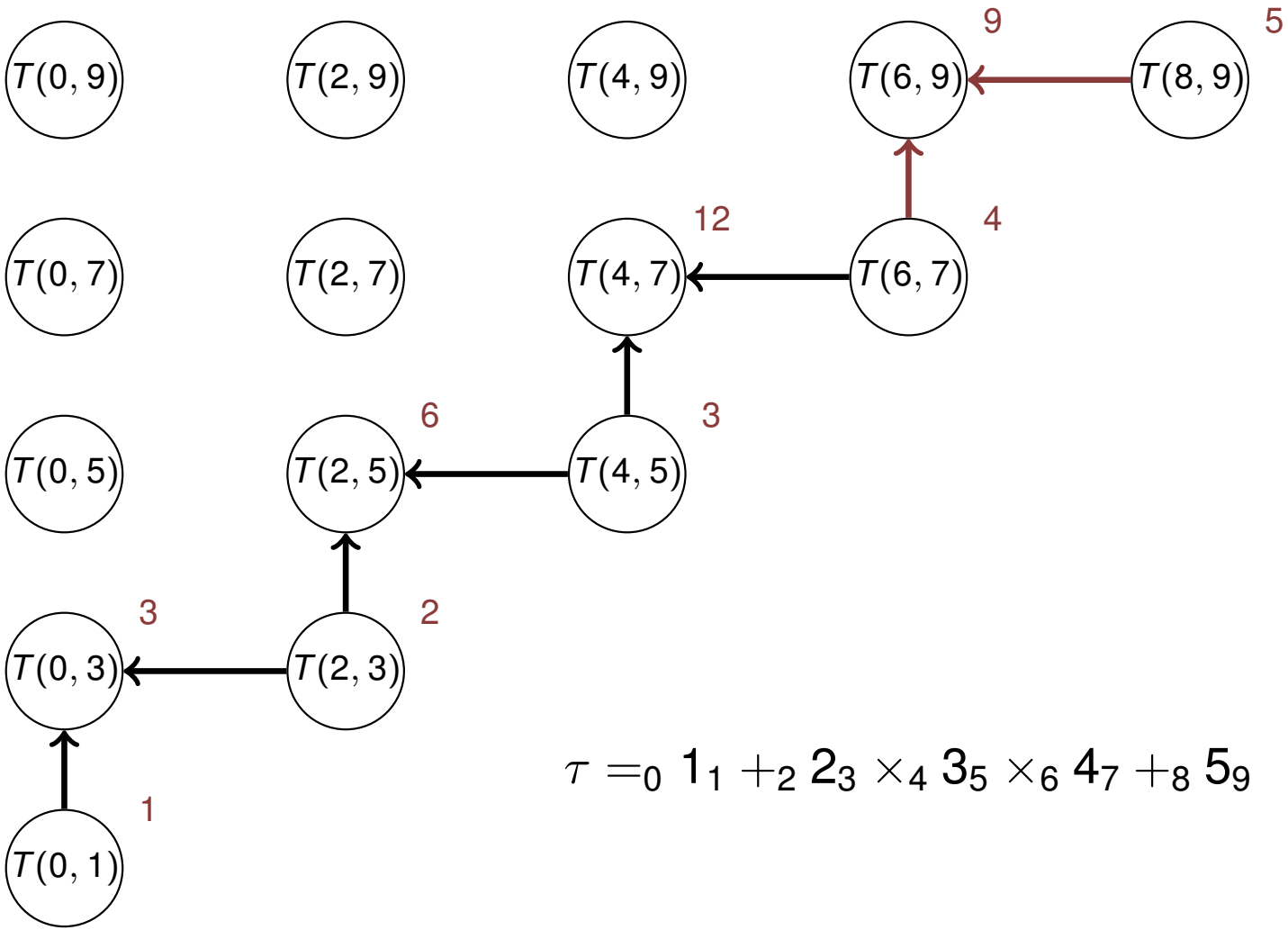
$$\tau =_0 1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



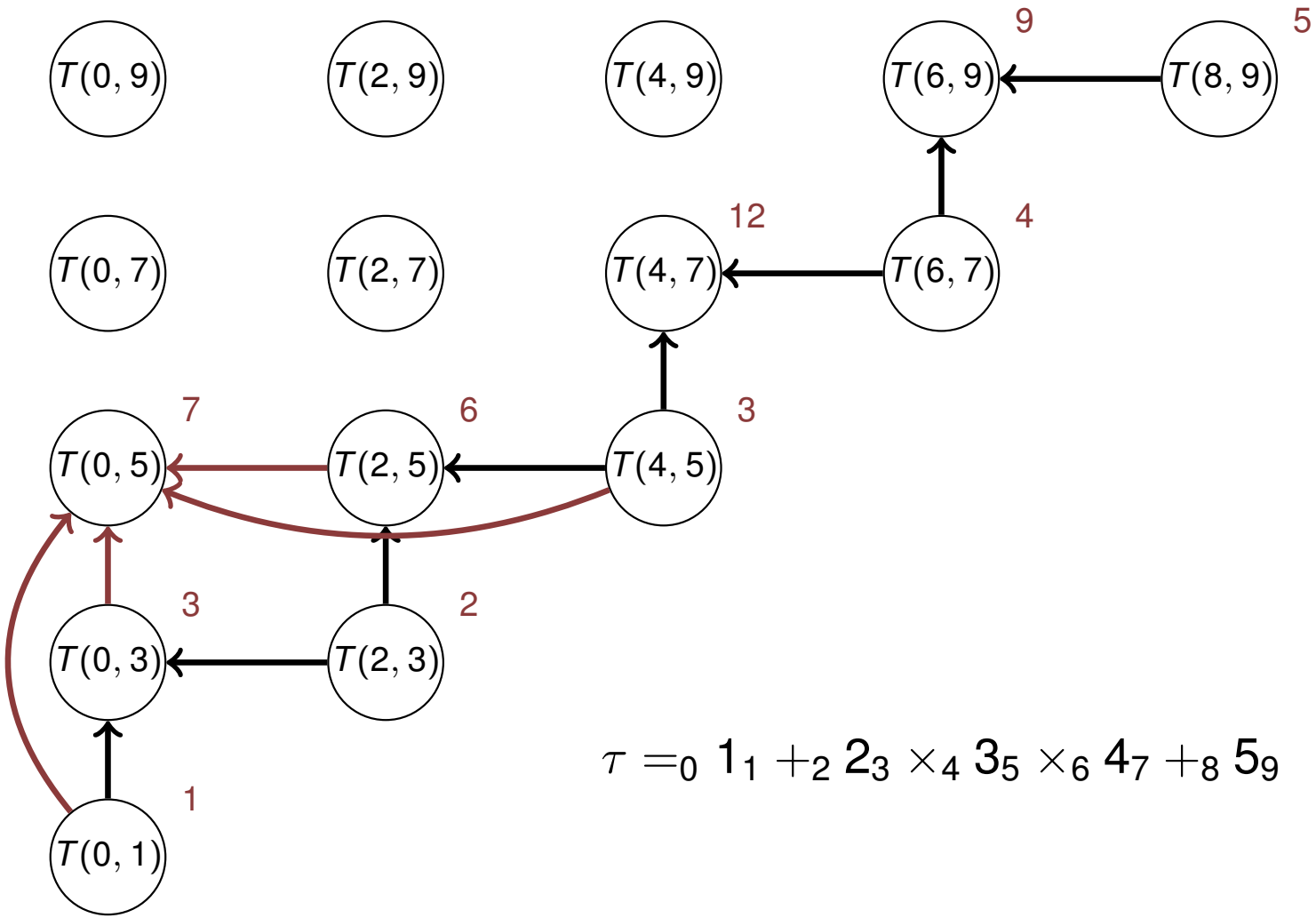
$$\tau = {}_0\,1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



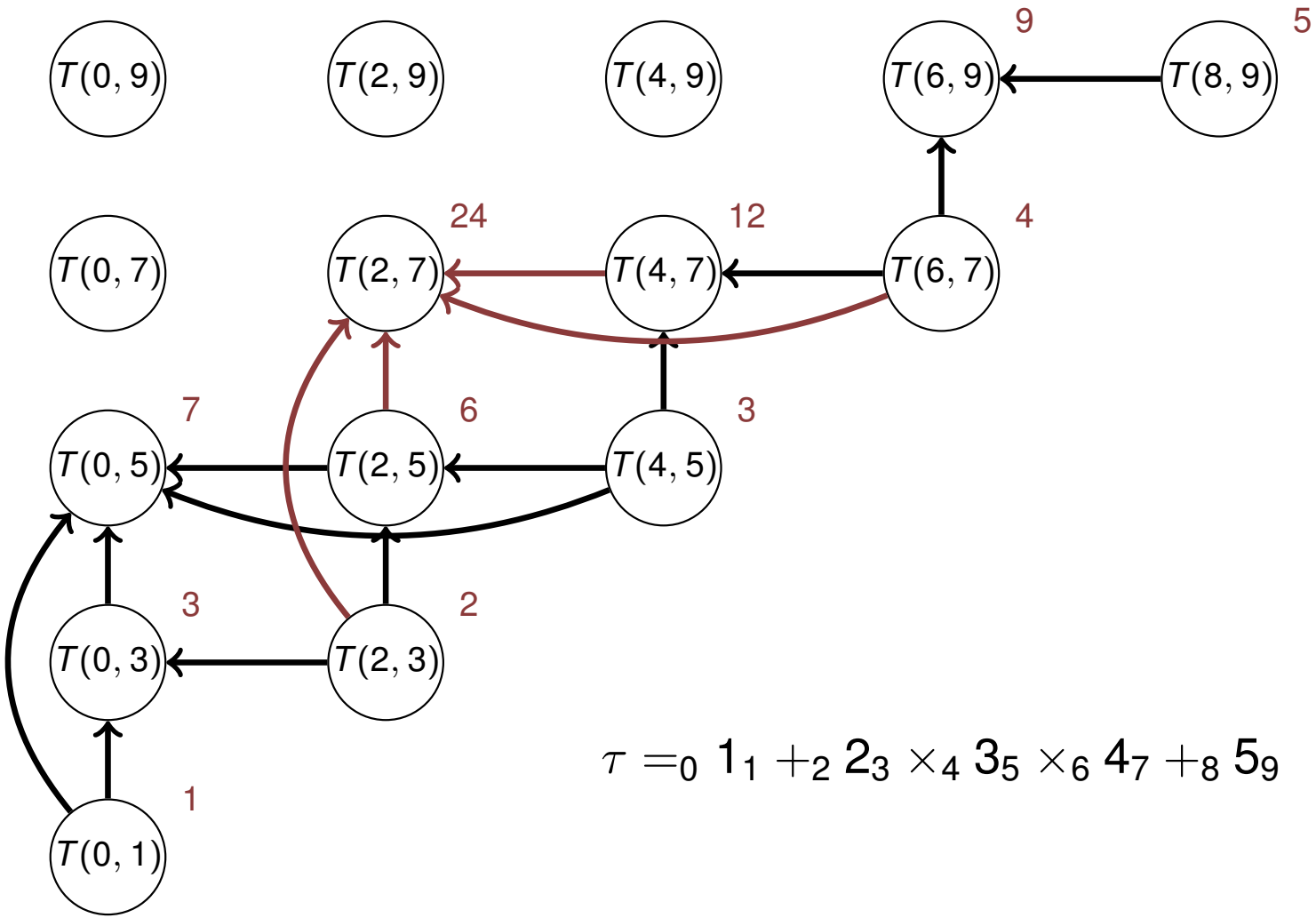
$$\tau = {}_0\,1_1 \,+_2\, 2_3 \,\times_4\, 3_5 \,\times_6\, 4_7 \,+_8\, 5_9$$

DÉROULEMENT



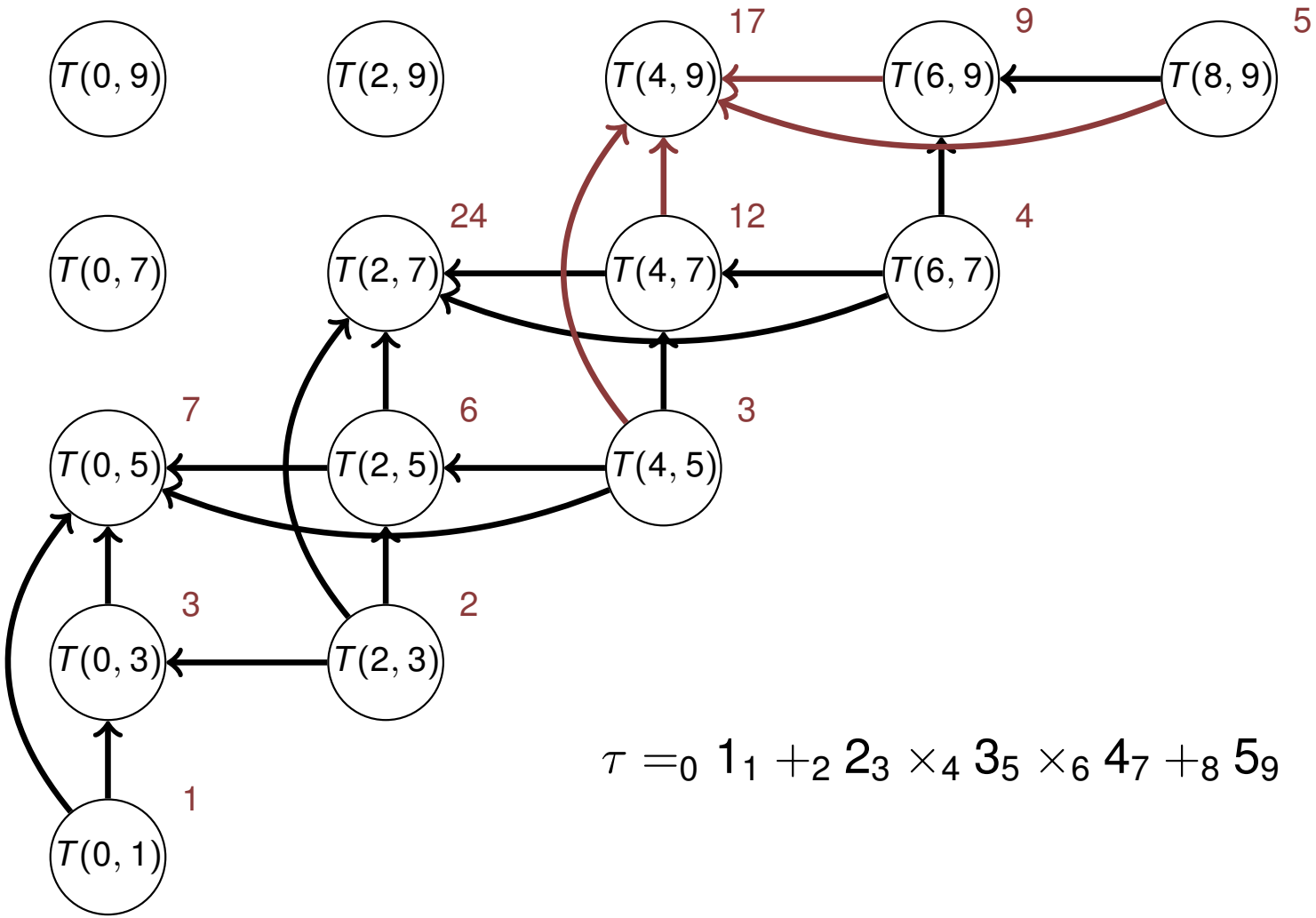
$$\tau = {}_0\,1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



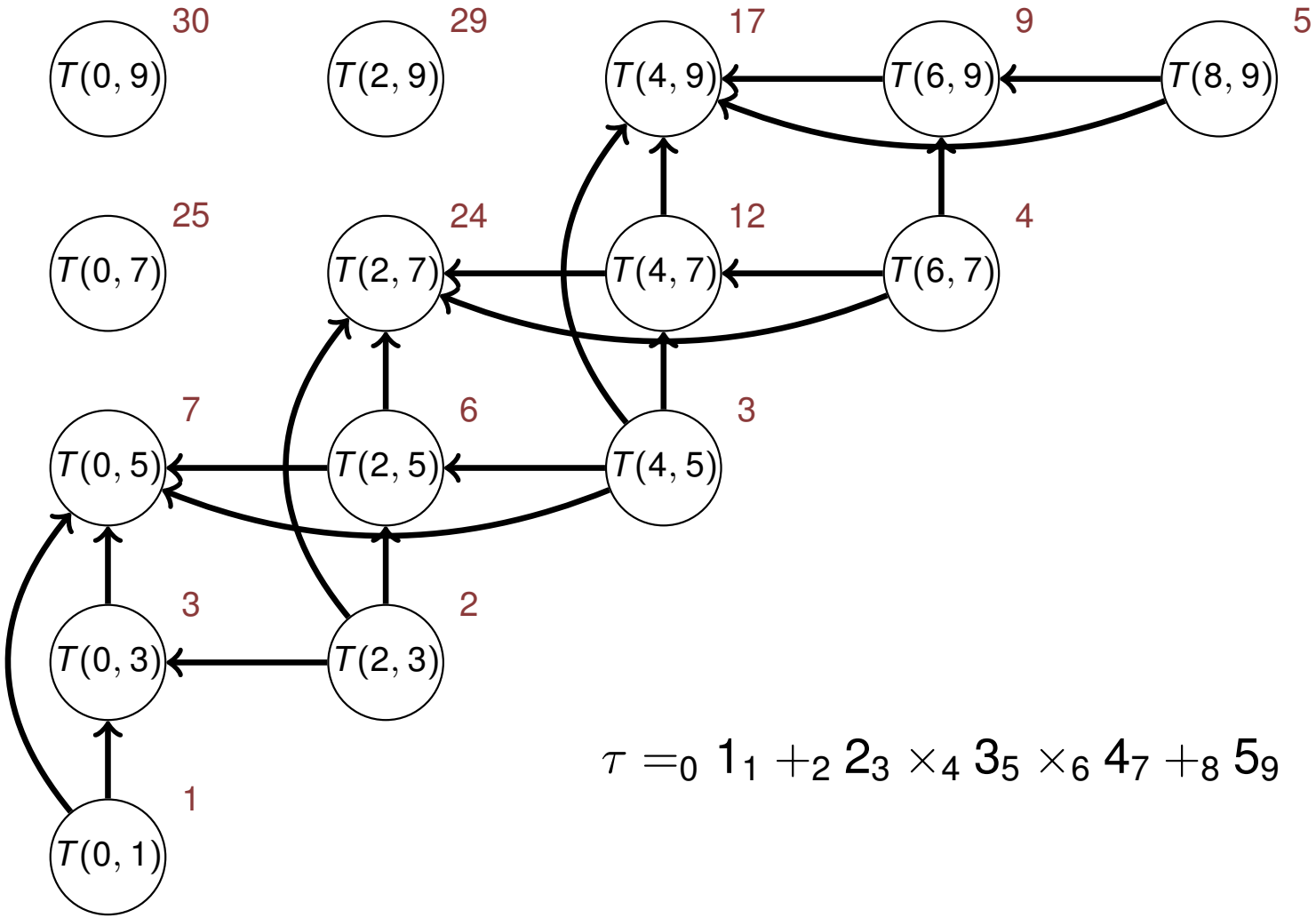
$$\tau = {}_0\,1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



$$\tau = {}_0\,1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

DÉROULEMENT



$$\tau = {}_0\,1_1 +_2 2_3 \times_4 3_5 \times_6 4_7 +_8 5_9$$

ENCORE PLUS FORT !

- On peut rajouter une variable pour tracer le chemin (et reconstruire comment la solution a été obtenue).
- On peut calculer de la même façon la valeur maximale [2].

  On trouve $T'(0, 9) = 81$.

  On notera ici que le chemin pour obtenir la solution n'est pas le même que dans le cas du minimum.

2. celle qu'a fournie le chamelier et qui lui a coûté la tête !

PROGRAMMATION DYNAMIQUE

I. LE PROBLÈME : trouver une meilleure solution

II. UNE HISTOIRE : Al Khwarismi

III. PRINCIPE : sous-optimalité, Richard Bellman

IV. SAC À DOS : un problème générique d'optimisation sous contrainte

V. SYNTHÈSE

UGA Université Grenoble Alpes

PROGRAMMATION DYNAMIQUE

Définition

La **programmation dynamique** est une méthode de décomposition d'un problème d'optimisation en sous-problèmes telle que la solution soit obtenue "facilement" dès que l'on connait celles de tous les sous-problèmes.
La difficulté ici est de déterminer les bons sous-problèmes.
Principe de (sous-optimalité) de Richard Bellman [a] : La solution d'un problème d'optimisation peut être obtenue par combinaisons de solutions optimales sur les sous-problèmes.

---

a. livre de R. Bellman "Dynamic Programming", Princeton (1953)

Cela ne vous rappelle rien ?

Attention : c'est différent du classique *divide-and-conquer* où il faut TOUT générer.
Exemples : TriFusion, enveloppe convexe, Karatzuba, Strassen, ...
De plus, cette technique impose que les sous-problèmes soient disjoints (partition).

Retour à Bagdad...

L'interprétation du principe de sous-optimalité de Bellman : comme les opérations $+$ et $\times$ sont monotones sur les entiers positifs, les expressions $x + y$ et $x \times y$ sont minimales (resp. maximales) lorsque x et y sont évaluées à leurs valeurs minimales (resp. maximales).

UGA Université Grenoble Alpes

RICHARD BELLMAN

Origine(s) du nom

Tiré de l'autobiographie de R. Bellman "Eye of the hurricane" en 1984 (qui coûte 374.98 US$ sur Amazon !).
Bellman travaillait à l'époque dans une entreprise aux US (RAND), il raconte qu'il a choisit **programming** pour ne pas effrayer son chef et parce que cela lui rappelait *Programmation Linéaire*.
Le terme **dynamic** vient de l'aspect de l'évolution de la méthode dans la remontée des différents étages de la récurrence.

1920-1984

from MacTutor History

Ce mathématicien (appliqué) américain est célèbre pour diverses contributions dans plusieurs domaines des mathématiques, il est surtout l'inventeur de la programmation dynamique, qui résolut à son époque de façon inespérée l'optimisation des sommes de fonctions monotones croissantes sous contraintes.
extrait de wikipedia

EXEMPLES

Pilzegal

Peut-on ramener le problème de *subset-sum* à un problème de programmation dynamique ?
Le problème est dans $\mathcal{NP}$ est-on réellement arrivé à le ramener dans $\mathcal{P}$?

Rendu de monnaie

Lorsque le système de pièces n'est pas canonique comment calculer la solution optimale ?
Pièces de valeur $v_1, \cdots, v_k, \cdots$
S somme à atteindre avec un minimum de pièces
Équation d'optimalité

$$\begin{aligned} V(S,k) &= \begin{cases} \min\{V(S,k-1), V(S-v_k,k)+1\} & \text{si } S \geqslant v_k \\ V(S,k-1) & \text{sinon} \end{cases} \\ V(0,0) &= 0 \end{aligned}$$

PROGRAMMATION DYNAMIQUE

I. LE PROBLÈME : trouver une meilleure solution

II. UNE HISTOIRE : Al Khwarismi

III. PRINCIPE : sous-optimalité, Richard Bellman

IV. SAC À DOS : un problème générique d'optimisation sous contrainte

V. SYNTHÈSE

LE PROBLÈME DU "*Sac à dos*"

Que fait un grenoblois lorsqu'il ne fait pas de l'Informatique ?
Informellement, le problème est de sélectionner des objets parmi un ensemble donné d'objets avec une *valeur* maximale sans dépasser la capacité du sac-à-dos.

Modélisation

- n objets ayant chacun un poids w_i et une valeur v_i ($1 \leqslant i \leqslant n$)
- un sac-à-dos S de capacité W.
- objectif : maximiser $V(S)$ la valeur totale du sac

$$V(S) = \sum_{i \in S} v_i \text{ sous la contrainte } \sum_{i \in S} w_i \leqslant W.$$

EXEMPLE

On considère l'instance suivante de 6 objets pour un sac de capacité 11 :

objet	poids	valeur
1	2	5
2	3	7
3	5	16
4	4	13
5	7	19
6	3	10

EXEMPLE

On considère l'instance suivante de 6 objets pour un sac de capacité 11 :

objet	poids	valeur
1	2	5
2	3	7
3	5	16
4	4	13
5	7	19
6	3	10

Résolution par un algorithme **glouton** :
On trie les objets par leurs rapports *qualité-prix* : $6, 4, 3, 5, 1, 2$ et on affecte les objets dans cet ordre tant qu'il y a de la place dans le sac.
Solution : 3 objets (6, 4 et 1) pour une valeur totale de 28.

SCHÉMA DE RÉSOLUTION

On note $V(k, p)$ la meilleure valeur totale des k premiers objets dans un sac de capacité p.

$$V(1,p) = \begin{cases} 0 & \text{si } p < w_1 \\ v_1 & \text{si } p \geqslant w_1 \end{cases}$$

$$V(k,p) = \begin{cases} V(k-1,p) & \text{si } p < w_k \\ \max\{V(k-1,p), V(k-1,p-w_k)+v_k\} & \text{si } p \geqslant w_k \end{cases}$$

La solution est $V(n, W)$

UGA Université Grenoble Alpes

Poids	Prix	0	1	2	3	4	5	6	7	8	9	10	11
2	5	0	0	5	5	5	5	5	5	5	5	5	5
3	7	0	0	5	7	7	12	12	12	12	12	12	12
5	16	0	0	5	7	7	16	16	21	23	23	28	28
4	13	0	0	5	7	13	16	18	21	23	29	29	34
7	19	0	0	5	7	13	16	18	21	23	29	29	34
3	10	0	0	5	10	13	16	18	23	26	29	31	34

PROGRAMMATION DYNAMIQUE

I. LE PROBLÈME : trouver une meilleure solution

II. UNE HISTOIRE : Al Khwarismi

III. PRINCIPE : sous-optimalité, Richard Bellman

IV. SAC À DOS : un problème générique d'optimisation sous contrainte

V. SYNTHÈSE

UGA Université Grenoble Alpes

SYNTHÈSE

Complexité

Le temps d'exécution de l'algorithme précédent pour résoudre le problème du sac-à-dos est dans $O(n \times W)$, il n'est plus seulement l-polynomial en n mais dépend du plus grand entier en entrée du problème (W).
Le problème est NP-difficile, mais il est *pseudo-polynomial* : Il peut être résolu efficacement quand les nombres w_i sont petits.

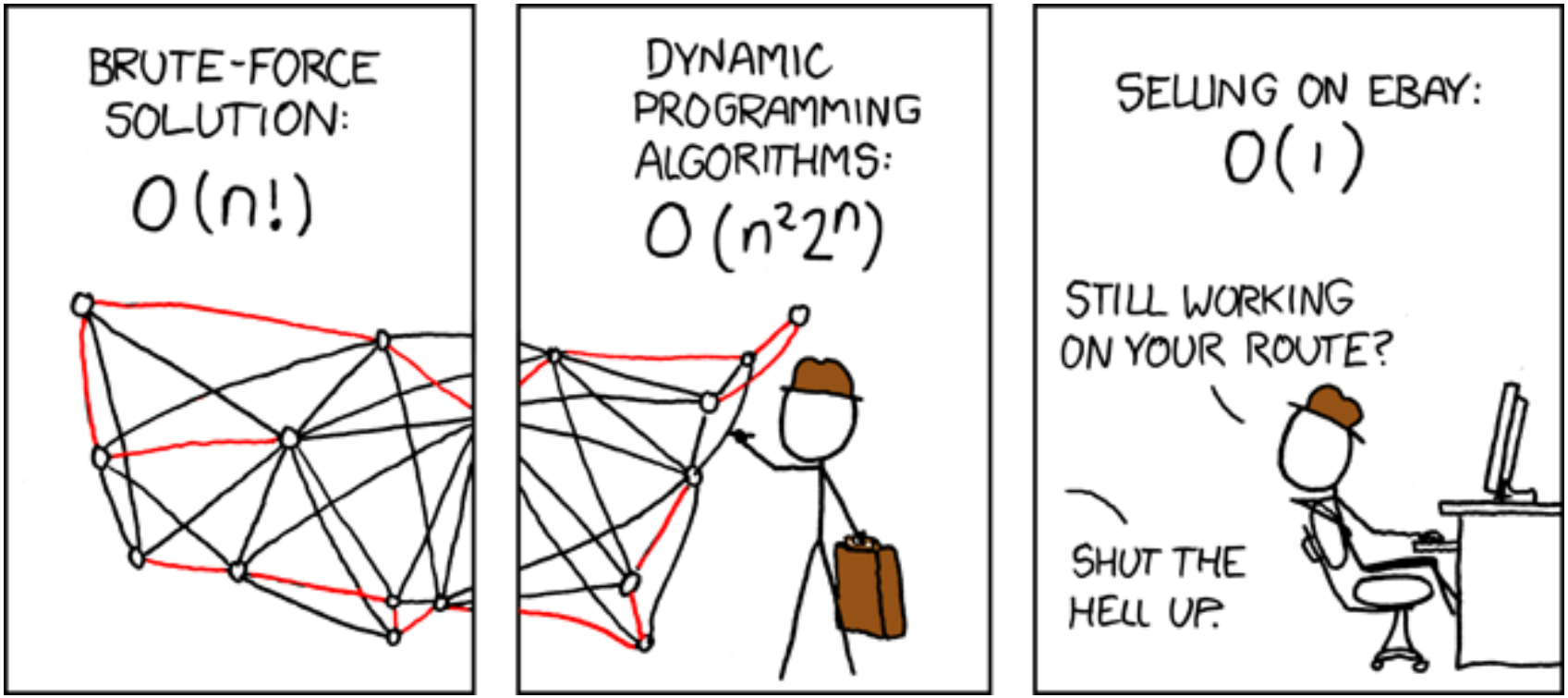
Take home message

- identifier une famille de sous-problèmes
- identifier les liens (structure) entre les sous-problèmes
- graphe sans circuit dans les dépendances des sous-problèmes
- donner une stratégie (ordre) pour le parcours du graphe

Approche récursive (top-down) : mémoïsation (éventuellement calculs redondants)
Approche itérative (bottom-up) : programmation dynamique (éventuellement calculs inutiles)

UGA Université Grenoble Alpes

TRAVELLING SALESMAN PROBLEM



Thanks XKCD https://xkcd.com/399/