

Chapitre 5

Programmation dynamique

HAI403I : Algorithmique 3

Université de Montpellier

1. Premier exemple : plus longue sous-suite croissante

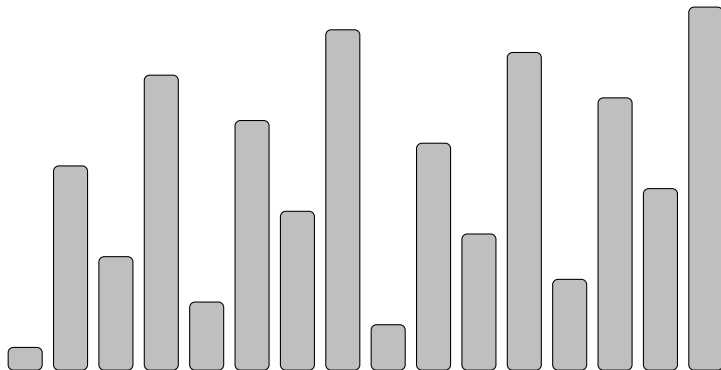
2. Qu'est-ce que la programmation dynamique ?

3. Deuxième exemple : choix de cours, le retour

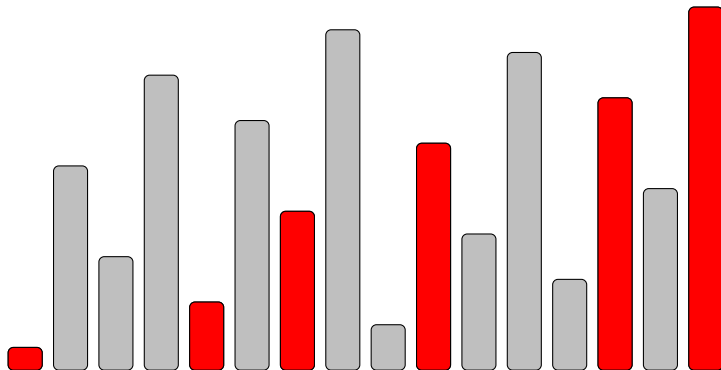
4. Troisième exemple : la distance d'édition

5. Exemple spécial : le voyageur de commerce

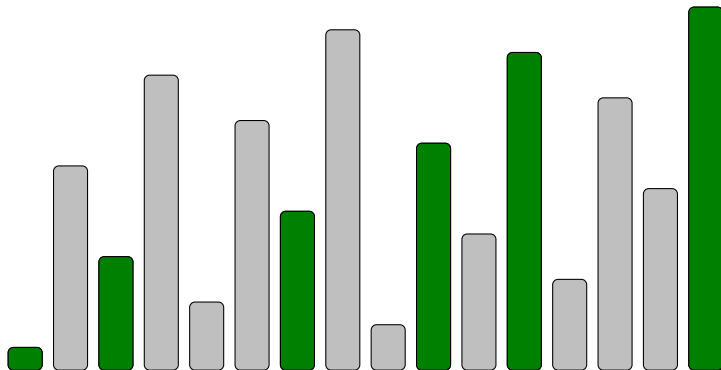
Plus longue sous-suite croissante



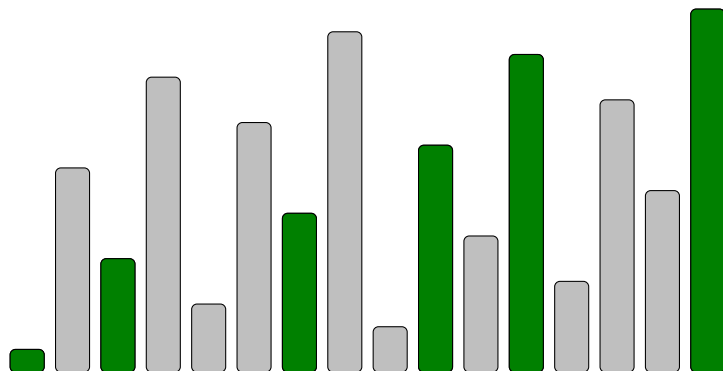
Plus longue sous-suite croissante



Plus longue sous-suite croissante



Plus longue sous-suite croissante



Définition

Une plus longue sous-suite croissante (plssc) d'un tableau T d'entiers est une suite la plus grande possible d'indices

$0 \leq i_1 < i_2 < \dots < i_k \leq n - 1$ telle que

$T[i_1] \leq T[i_2] \leq \dots \leq T[i_k]$.

Formalisation du problème

Entrée Un tableau T de n entiers

Sortie 1 Une PLSSC de T

Formalisation du problème

Entrée Un tableau T de n entiers

Sortie 1 Une PLSSC de T

Sortie 2 La longueur d'une PLSSC de T

Formalisation du problème

Entrée Un tableau T de n entiers

Sortie 1 Une PLSSC de T

Sortie 2 La longueur d'une PLSSC de T

Exemple précédent

Entrée $T = [1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8, 16]$

Sortie 1 $[1, 3, 7, 10, 12, 16]$ ou $[1, 5, 7, 10, 14, 16]$ ou ...

Sortie 2 6

Formalisation du problème

Entrée Un tableau T de n entiers

Sortie 1 Une PLSSC de T

Sortie 2 La longueur d'une PLSSC de T

Exemple précédent

Entrée $T = [1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8, 16]$

Sortie 1 $[1, 3, 7, 10, 12, 16]$ ou $[1, 5, 7, 10, 14, 16]$ ou ...

Sortie 2 6

- ▶ Algo. naïf : considérer toutes les sous-suites $\rightsquigarrow O(2^n)$

Formalisation du problème

Entrée Un tableau T de n entiers

Sortie 1 Une PLSSC de T

Sortie 2 La longueur d'une PLSSC de T

Exemple précédent

Entrée $T = [1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8, 16]$

Sortie 1 $[1, 3, 7, 10, 12, 16]$ ou $[1, 5, 7, 10, 14, 16]$ ou ...

Sortie 2 6

- ▶ Algo. naïf : considérer toutes les sous-suites $\rightsquigarrow O(2^n)$

Algorithme de complexité polynomiale ?

Formule récursive pour PLSSC

- ▶ $\text{plssc}(T)$: longueur des PLSSC de T
- ▶ $L[i]$: longueur des PLSSC de T finissant en case $T[i]$

Formule récursive pour PLSSC

- ▶ $\text{plssc}(T)$: longueur des PLSSC de T
- ▶ $L[i]$: longueur des PLSSC de T finissant en case $T[i]$

Remarque

$$\text{plssc}(T) = \max_{0 \leq i < n} L[i]$$

Formule récursive pour PLSSC

- ▶ $\text{plssc}(T)$: longueur des PLSSC de T
- ▶ $L[i]$: longueur des PLSSC de T finissant en case $T[i]$

Remarque

$$\text{plssc}(T) = \max_{0 \leq i < n} L[i]$$

Lemme

$$L[i] = \begin{cases} 1 & \text{si } i = 0 \\ 1 + \max\{L[j] : j < i \text{ et } T[j] \leq T[i]\} & \text{pour } 1 \leq i < n \end{cases}$$

Formule récursive pour PLSSC

- ▶ $\text{plssc}(T)$: longueur des PLSSC de T
- ▶ $L[i]$: longueur des PLSSC de T finissant en case $T[i]$

Remarque

$$\text{plssc}(T) = \max_{0 \leq i < n} L[i]$$

Lemme

$$L[i] = \begin{cases} 1 & \text{si } i = 0 \\ 1 + \max\{L[j] : j < i \text{ et } T[j] \leq T[i]\} & \text{pour } 1 \leq i < n \end{cases}$$

Preuve On montre deux inégalités :

- \geq si $L[j_m]$ est le max, il existe une SSC $T[i_1], \dots, T[i_k]$ de longueur $L[j_m] \rightsquigarrow T[i_1], \dots, T[i_k], T[i]$ est une SSC de longueur $1 + L[j_m]$
- \leq si $T[i_1], \dots, T[i_k]$ est une PLSSC finissant en $i (= i_k)$, alors $L[i_{k-1}] \geq k - 1$ donc $\max\{L[j] : \dots\} \geq k - 1$

Formule récursive pour PLSSC

- ▶ $\text{plssc}(T)$: longueur des PLSSC de T
- ▶ $L[i]$: longueur des PLSSC de T finissant en case $T[i]$

Remarque

$$\text{plssc}(T) = \max_{0 \leq i < n} L[i]$$

Lemme

$$L[i] = \begin{cases} 1 & \text{si } i = 0 \\ 1 + \max\{L[j] : j < i \text{ et } T[j] \leq T[i]\} & \text{pour } 1 \leq i < n \end{cases}$$

Ne pas programmer la formule par un algorithme récursif !

Algorithme PLSSC

Algorithme : PLSSC(T)

$L \leftarrow$ tableau de taille n , initialisé à 1

$M \leftarrow L[0]$ // M contient $\max_i L[i]$

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$ // m contient $\max\{L[j] : j < i, T[j] \leq T[i]\}$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$

$L[i] \leftarrow 1 + m$

 si $L[i] > M$ alors $M \leftarrow L[i]$

retourner M

Correction et complexité

Théorème

L'algorithme PLSSC calcule $\text{plssc}(T)$ en temps $O(n^2)$.

Algorithme : $\text{PLSSC}(T)$

$L \leftarrow$ tableau de taille n , initialisé à 1

$M \leftarrow L[0]$

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$

$L[i] \leftarrow 1 + m$

 si $L[i] > M$ alors $M \leftarrow L[i]$

retourner M

Correction et complexité

Théorème

L'algorithme PLSSC calcule $\text{plssc}(T)$ en temps $O(n^2)$.

Algorithme : $\text{PLSSC}(T)$

$L \leftarrow$ tableau de taille n , initialisé à 1

$M \leftarrow L[0]$

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$

$L[i] \leftarrow 1 + m$

 si $L[i] > M$ alors $M \leftarrow L[i]$

retourner M

Preuve de correction : utilisation de la formule récursive

Preuve de complexité : double boucle

Correction et complexité

Théorème

L'algorithme PLSSC calcule $\text{plssc}(T)$ en temps $O(n^2)$.

Algorithme : $\text{PLSSC}(T)$

$L \leftarrow$ tableau de taille n , initialisé à 1

$M \leftarrow L[0]$

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$

$L[i] \leftarrow 1 + m$

 si $L[i] > M$ alors $M \leftarrow L[i]$

retourner M

Preuve de correction : utilisation de la formule récursive

Preuve de complexité : double boucle

Comment calculer une plssc (en plus de sa longueur) ?

- ▶ Retenir les indices des max
- ▶ Reconstruire *a posteriori*

Algorithme PLSSC avec reconstruction

Algorithme : PLSSC(T)

$L \leftarrow$ tableau de taille n , initialisé à 1

$Ind \leftarrow$ tab de taille n , initialisé à -1

$i_M \leftarrow 0$ // i_M est l'indice de fin
d'une PLSSC

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$; $Ind[i] \leftarrow j$

$L[i] \leftarrow 1 + m$

 si $L[i] > L[i_M]$ alors $i_M \leftarrow i$

retourner $L[i_M]$, i_M et Ind

Algorithme PLSSC avec reconstruction

Algorithme : PLSSC(T)

$L \leftarrow$ tableau de taille n , initialisé à 1

$Ind \leftarrow$ tab de taille n , initialisé à -1

$i_M \leftarrow 0$ // i_M est l'indice de fin
d'une PLSSC

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$; $Ind[i] \leftarrow j$

$L[i] \leftarrow 1 + m$

 si $L[i] > L[i_M]$ alors $i_M \leftarrow i$

retourner $L[i_M]$, i_M et Ind

Algorithme :

PLSSC_REC(T, i_M, Ind)

$Seq \leftarrow$ tab de taille n ,
initialisé à 0

$i \leftarrow i_M$

tant que $i \neq -1$ faire

$Seq[i] \leftarrow 1$

$i \leftarrow Ind[i]$

retourner Seq

Algorithme PLSSC avec reconstruction

Algorithme : PLSSC(T)

$L \leftarrow$ tableau de taille n , initialisé à 1

$Ind \leftarrow$ tab de taille n , initialisé à -1

$i_M \leftarrow 0$ // i_M est l'indice de fin
d'une PLSSC

pour $i = 1$ à $n - 1$ faire

$m \leftarrow 0$

 pour $j = 0$ à $i - 1$ faire

 si $T[j] \leq T[i]$ et $L[j] > m$ alors

$m \leftarrow L[j]$; $Ind[i] \leftarrow j$

$L[i] \leftarrow 1 + m$

 si $L[i] > L[i_M]$ alors $i_M \leftarrow i$

retourner $L[i_M]$, i_M et Ind

Algorithme :

PLSSC_REC(T, i_M, Ind)

$Seq \leftarrow$ tab de taille n ,
initialisé à 0

$i \leftarrow i_M$

tant que $i \neq -1$ faire

$Seq[i] \leftarrow 1$

$i \leftarrow Ind[i]$

retourner Seq

Lemme

L'algo. PLSSC_REC reconstruit une PLSSC de T en temps $O(n)$.

1. Premier exemple : plus longue sous-suite croissante
2. Qu'est-ce que la programmation dynamique ?
3. Deuxième exemple : choix de cours, le retour
4. Troisième exemple : la distance d'édition
5. Exemple spécial : le voyageur de commerce

Idée générale

Programmation dynamique = récursion sans répétition

Idée générale

Programmation dynamique = récursion sans répétition

Ingrédients

1. Formule récursive pour la valeur optimale
 - ▶ en fonction des valeurs de sous-problèmes
 - ▶ sous-problèmes possiblement nombreux et non disjoints

Idée générale

Programmation dynamique = récursion sans répétition

Ingrédients

1. Formule récursive pour la valeur optimale
 - ▶ en fonction des valeurs de sous-problèmes
 - ▶ sous-problèmes possiblement nombreux et non disjoints
2. Algorithme itératif pour la valeur optimale
 - ▶ en commençant par les plus petits sous-problèmes
 - ▶ approche « *bottom-up* »

Idée générale

Programmation dynamique = récursion sans répétition

Ingrédients

1. Formule récursive pour la valeur optimale
 - ▶ en fonction des valeurs de sous-problèmes
 - ▶ sous-problèmes possiblement nombreux et non disjoints
2. Algorithme itératif pour la valeur optimale
 - ▶ en commençant par les plus petits sous-problèmes
 - ▶ approche « *bottom-up* »
3. Reconstruction de la solution *a posteriori*
 - ▶ ajout d'informations à l'algo. pour la valeur
 - ▶ algorithme de reconstruction indépendant

Idée générale

Programmation dynamique = récursion sans répétition

Ingrédients

1. Formule récursive pour la valeur optimale
 - ▶ en fonction des valeurs de sous-problèmes
 - ▶ sous-problèmes possiblement nombreux et non disjoints
2. Algorithme itératif pour la valeur optimale
 - ▶ en commençant par les plus petits sous-problèmes
 - ▶ approche « *bottom-up* »
3. Reconstruction de la solution *a posteriori*
 - ▶ ajout d'informations à l'algo. pour la valeur
 - ▶ algorithme de reconstruction indépendant

« Diviser pour régner »

Sous-problèmes disjoints, approche « *top-down* »

Ingrédient 1 : Formule récursive

Partie la plus importante (et difficile) !

Ingrédient 1 : Formule récursive

Partie la plus importante (et difficile) !

Étapes

1. Spécification précise du problème
2. Formule récursive basée sur les solutions d'instances plus petites du même problème exactement

Ingrédient 1 : Formule récursive

Partie la plus importante (et difficile) !

Étapes

1. Spécification précise du problème
2. Formule récursive basée sur les solutions d'instances plus petites du même problème exactement

Plus longue sous-suite croissante

1. Définition de $L[i]$ en pas seulement $\text{plssc}(T)$
2. Expression de $L[i]$ en fonction des $L[j]$, $j < i$

Ingrédient 1 : Formule récursive

Partie la plus importante (et difficile) !

Étapes

1. Spécification précise du problème
2. Formule récursive basée sur les solutions d'instances plus petites du même problème exactement

Plus longue sous-suite croissante

1. Définition de $L[i]$ en pas seulement $\text{plssc}(T)$
2. Expression de $L[i]$ en fonction des $L[j]$, $j < i$

En pratique, étape souvent (très) guidée

Ingrédient 2 : Algorithme itératif

Partie plutôt facile... mais attention quand même !

Ingrédient 2 : Algorithme itératif

Partie plutôt facile... mais attention quand même !

Étapes

1. choix d'une structure de données (*très* souvent un tableau)
2. ordre de calcul si tableau multi-dimensionnel *cf. ex. suivants*
3. écriture effective de l'algorithme
4. analyse de complexité

Ingrédient 2 : Algorithme itératif

Partie plutôt facile... mais attention quand même !

Étapes

1. choix d'une structure de données (*très souvent un tableau*)
2. ordre de calcul si tableau multi-dimensionnel *cf. ex. suivants*
3. écriture effective de l'algorithme
4. analyse de complexité

Plus longue sous-suite croissante

1. Tableau L
2. Ordre croissant

Ingrédient 2 : Algorithme itératif

Partie plutôt facile... mais attention quand même !

Étapes

1. choix d'une structure de données (*très souvent un tableau*)
2. ordre de calcul si tableau multi-dimensionnel *cf. ex. suivants*
3. écriture effective de l'algorithme
4. analyse de complexité

Plus longue sous-suite croissante

1. Tableau L
2. Ordre croissant

En pratique, étape souvent non guidée

Ingrédient 3 : Reconstruction

Partie de difficulté très variable !

Ingrédient 3 : Reconstruction

Partie de difficulté très variable !

Étapes

1. ajout d'informations supplémentaires à l'algo. précédent
2. *redescende* depuis la solution générale vers les instances petites

Ingrédient 3 : Reconstruction

Partie de difficulté très variable !

Étapes

1. ajout d'informations supplémentaires à l'algo. précédent
2. *redescente* depuis la solution générale vers les instances petites

Plus longue sous-suite croissante

1. tableau Ind , indice i_M
2. descente depuis $T[i_M]$, en suivant Ind

Ingrédient 3 : Reconstruction

Partie de difficulté très variable !

Étapes

1. ajout d'informations supplémentaires à l'algo. précédent
2. *redescente* depuis la solution générale vers les instances petites

Plus longue sous-suite croissante

1. tableau Ind , indice i_M
2. descente depuis $T[i_M]$, en suivant Ind

En pratique, étape pas toujours effectuée

Conclusion sur la programmation dynamique

Comparaison avec les algorithmes gloutons :

- ▶ Algo. glouton : cas particulier de programmation dynamique avec un seul sous-problème
- ▶ Souvent pas suffisant

Conclusion sur la programmation dynamique

Comparaison avec les algorithmes gloutons :

- ▶ Algo. glouton : cas particulier de programmation dynamique avec un seul sous-problème
- ▶ Souvent pas suffisant

Les algorithmes gloutons fonctionnent rarement !

Conclusion sur la programmation dynamique

Comparaison avec les algorithmes gloutons :

- ▶ Algo. glouton : cas particulier de programmation dynamique avec un seul sous-problème
- ▶ Souvent pas suffisant

Les algorithmes gloutons fonctionnent rarement !

D'où vient ce nom ?

- ▶ Belman (1940) : travaux en optimisation mathématique
 - ▶ Programmation : planification, ordonnancement
 - ▶ Dynamique : « *it's impossible to use the word dynamic in a pejorative sense* »
- ▶ Origine du mot peu claire : référence à la programmation linéaire, et/ou problèmes de financements (cf Wikipédia)

1. Premier exemple : plus longue sous-suite croissante
2. Qu'est-ce que la programmation dynamique ?
3. Deuxième exemple : choix de cours, le retour
4. Troisième exemple : la distance d'édition
5. Exemple spécial : le voyageur de commerce

Revisitons nos classiques

Choix de cours valué

Entrée cours $C_i = (d_i, f_i, e_i)$ [début, fin, crédits ECTS], $0 \leq i < n$

Sortie un sous-ensemble de cours compatibles qui maximise le nombre total de crédits ECTS

Revisitons nos classiques

Choix de cours valué

Entrée cours $C_i = (d_i, f_i, e_i)$ [début, fin, crédits ECTS], $0 \leq i < n$

Sortie un sous-ensemble de cours compatibles qui maximise le nombre total de crédits ECTS

Lemme

L'algorithme CHOIXCOURSGLOUTON vu pour le choix de cours non valué est arbitrairement mauvais sur ce problème !

Revisitons nos classiques

Choix de cours valué

Entrée cours $C_i = (d_i, f_i, e_i)$ [début, fin, crédits ECTS], $0 \leq i < n$

Sortie un sous-ensemble de cours compatibles qui maximise le nombre total de crédits ECTS

Lemme

L'algorithme CHOIXCOURSGLOUTON vu pour le choix de cours non valué est arbitrairement mauvais sur ce problème !

Exemple

$C = \{(9, 12, 1), (11, 14, 100), (13, 15, 1)\}$:

- ▶ CHOIXCOURSGLOUTON renvoie une solution à 2 ECTS
- ▶ L'optimal est 50 fois meilleur !

Formule récursive pour le choix de cours valué

Cours triés par ordre de fin croissante C_0, C_1, \dots, C_{n-1}

Notations

- ▶ $\text{pred}(k) = \max\{j : f_j \leq d_k\}$ (avec $\max(\emptyset) = -1$)
- ▶ $\text{maxECTS}(k)$: nombre maximal d'ECTS avec les cours C_0, \dots, C_k ($\text{maxECTS}(-1) = 0$)

Formule récursive pour le choix de cours valué

Cours triés par ordre de fin croissante C_0, C_1, \dots, C_{n-1}

Notations

- ▶ $\text{pred}(k) = \max\{j : f_j \leq d_k\}$ (avec $\max(\emptyset) = -1$)
- ▶ $\text{maxECTS}(k)$: nombre maximal d'ECTS avec les cours C_0, \dots, C_k ($\text{maxECTS}(-1) = 0$)

Lemme

$\text{maxECTS}(0) = e_0$ et pour $1 \leq k < n$,

$\text{maxECTS}(k) = \max(\text{maxECTS}(k-1), e_k + \text{maxECTS}(\text{pred}(k)))$

Formule récursive pour le choix de cours valué

Cours triés par ordre de fin croissante C_0, C_1, \dots, C_{n-1}

Notations

- ▶ $\text{pred}(k) = \max\{j : f_j \leq d_k\}$ (avec $\max(\emptyset) = -1$)
- ▶ $\text{maxECTS}(k)$: nombre maximal d'ECTS avec les cours C_0, \dots, C_k ($\text{maxECTS}(-1) = 0$)

Lemme

$\text{maxECTS}(0) = e_0$ et pour $1 \leq k < n$,

$\text{maxECTS}(k) = \max(\text{maxECTS}(k-1), e_k + \text{maxECTS}(\text{pred}(k)))$

Preuve : la solution optimale pour C_0, \dots, C_k contient-elle C_k ?

- ▶ Si oui, les autres cours choisis sont parmi $C_0, \dots, C_{\text{pred}(k)}$ \rightsquigarrow nb d'ECTS : $e_k + \text{maxECTS}(\text{pred}(k))$
- ▶ Si non, nb d'ECTS : $\text{maxECTS}(k-1)$

Algorithme pour le choix de cours valué

Algorithme : MAXECTS(C)

Trier C par dates de fin croissantes

$P \leftarrow$ tableau de taille n , initialisé à -1 // prédécesseurs

pour $k = 1$ à $n - 1$ faire

 pour $j = 0$ à $k - 1$ faire

 si $f_j \leq d_k$ alors $P[k] \leftarrow j$

$M \leftarrow$ tableau de taille n , initialisé à 0

$M[0] \leftarrow e_0$

pour $k = 1$ à $n - 1$ faire

 si $P[k] \neq -1$ alors $M[k] \leftarrow \max(M[k - 1], e_k + M[P[k]])$

 sinon $M[k] \leftarrow \max(M[k - 1], e_k)$

retourner $M[n - 1]$

Algorithme pour le choix de cours valué

Algorithme : MAXECTS(C)

Trier C par dates de fin croissantes

$P \leftarrow$ tableau de taille n , initialisé à -1 // prédécesseurs

pour $k = 1$ à $n - 1$ faire

 pour $j = 0$ à $k - 1$ faire

 si $f_j \leq d_k$ alors $P[k] \leftarrow j$

$M \leftarrow$ tableau de taille n , initialisé à 0

$M[0] \leftarrow e_0$

pour $k = 1$ à $n - 1$ faire

 si $P[k] \neq -1$ alors $M[k] \leftarrow \max(M[k - 1], e_k + M[P[k]])$

 sinon $M[k] \leftarrow \max(M[k - 1], e_k)$

retourner $M[n - 1]$

Complexité : $O(n^2)$ (calcul de P , calcul de M en $O(n)$)

Et pour le sac-à-dos ?

Sac-à-dos (non fractionnaire)

Entrée un ensemble d'objets, ayant une taille t_i et une valeur v_i
une taille T de sac-à-dos

Sortie un sous-ensemble d'objets qui rentre dans le sac, et maximise
la valeur totale

Et pour le sac-à-dos ?

Sac-à-dos (non fractionnaire)

Entrée un ensemble d'objets, ayant une taille t_i et une valeur v_i
une taille T de sac-à-dos

Sortie un sous-ensemble d'objets qui rentre dans le sac, et maximise
la valeur totale

Lemme

*L'algorithme SÀDFRACGLOUTON n'est pas optimal pour le
sac-à-dos non fractionnaire.*

Et pour le sac-à-dos ?

Sac-à-dos (non fractionnaire)

Entrée un ensemble d'objets, ayant une taille t_i et une valeur v_i ;
une taille T de sac-à-dos

Sortie un sous-ensemble d'objets qui rentre dans le sac, et maximise
la valeur totale

Lemme

*L'algorithme SÀDFRACGLOUTON n'est pas optimal pour le
sac-à-dos non fractionnaire.*

- ▶ Exemples vus en cours et TD
- ▶ Algos de programmation dynamique optimaux \rightsquigarrow cf. TD

1. Premier exemple : plus longue sous-suite croissante
2. Qu'est-ce que la programmation dynamique ?
3. Deuxième exemple : choix de cours, le retour
4. Troisième exemple : la distance d'édition
5. Exemple spécial : le voyageur de commerce

Corrigeons les erreurs

AGORRYTNES

Corrigeons les erreurs

↓ ↓
AGORRYTNES

Corrigeons les erreurs

↓ ↓
AGORRYTNES

ALGORITHMES
x

Corrigeons les erreurs

↓ ↓
AGORRYTNES

ALGORITHMES
x

À quelle distance se trouve-t-on du mot correct ?

Corrigeons les erreurs

AGORRYTNES
ALGORITHMES

À quelle distance se trouve-t-on du mot correct ?

Distance 5

AGORRYTNES → ALGORRYTNES → ALGORYTNES
↓
ALGORITHMES ← ALGORITHNES ← ALGORITNES

La distance d'édition

Définition

La distance d'édition (ou de Levenshtein, ou d'Ulam) entre deux mots A et B est la longueur de la plus courte suite de transformations pour passer de A à B , avec les transformations suivantes :

- ▶ insertion d'une nouvelle lettre
- ▶ suppression d'une lettre
- ▶ remplacement d'une lettre par une autre

La distance d'édition

Définition

La distance d'édition (ou de Levenshtein, ou d'Ulam) entre deux mots A et B est la longueur de la plus courte suite de transformations pour passer de A à B , avec les transformations suivantes :

- ▶ insertion d'une nouvelle lettre
- ▶ suppression d'une lettre
- ▶ remplacement d'une lettre par une autre

Définition alternative

Nombre minimal de désaccords dans un alignement de A et B

A	_	G	O	R	R	Y	T	_	N	E	S
A	L	G	O	_	R	I	T	H	M	E	S

Définition du problème

Entrée Deux mots A et B sur un alphabet
(mot : chaîne de caractère ou tableau de caractères ou ...)

Sortie 1 La distance d'édition entre A et B

Sortie 2 Une suite de transformations minimale de A à B

Définition du problème

Entrée Deux mots A et B sur un alphabet
(mot : chaîne de caractère ou tableau de caractères ou ...)

Sortie 1 La distance d'édition entre A et B

Sortie 2 Une suite de transformations minimale de A à B

Utilité

- ▶ Orthographe :
 - ▶ Correcteur orthographique
 - ▶ Reconnaissance optique de caractères
- ▶ Linguistique (proximité de langues)
- ▶ Bioinformatique :
 - ▶ similarité de séquences ADN
 - ▶ similarité d'arbres phylogénétiques
- ▶ ...

Formalisation

- ▶ $\delta(A, B)$: distance entre A et B
- ▶ $\text{edit}(i, j)$: distance entre $A[0, i]$ et $B[0, j]$
- ▶ convention : $\text{edit}(i, -1) = i + 1$ et $\text{edit}(-1, j) = j + 1$
pourquoi ?

Formalisation

- ▶ $\delta(A, B)$: distance entre A et B
- ▶ $\text{edit}(i, j)$: distance entre $A[0, i]$ et $B[0, j]$
- ▶ convention : $\text{edit}(i, -1) = i + 1$ et $\text{edit}(-1, j) = j + 1$
pourquoi ?

si $|A| = m$ et $|B| = n$, $\delta(A, B) = \text{edit}(m - 1, n - 1)$

Formalisation

- ▶ $\delta(A, B)$: distance entre A et B
- ▶ $\text{edit}(i, j)$: distance entre $A[0, i]$ et $B[0, j]$
- ▶ convention : $\text{edit}(i, -1) = i + 1$ et $\text{edit}(-1, j) = j + 1$
pourquoi ?

si $|A| = m$ et $|B| = n$, $\delta(A, B) = \text{edit}(m - 1, n - 1)$

Trois alignements possibles

$A[0, i-1]$		$A[0, i]$		$A[0, i-1]$	
\dots		\dots		\dots	
\dots	$A[i]$	\dots	$-$	\dots	$A[i]$
$B[0, j]$	$-$	$B[0, j-1]$	$B[j]$	$B[0, j-1]$	$B[j]$

Formalisation

- ▶ $\delta(A, B)$: distance entre A et B
- ▶ $\text{edit}(i, j)$: distance entre $A[0, i]$ et $B[0, j]$
- ▶ convention : $\text{edit}(i, -1) = i + 1$ et $\text{edit}(-1, j) = j + 1$
pourquoi ?

si $|A| = m$ et $|B| = n$, $\delta(A, B) = \text{edit}(m - 1, n - 1)$

Trois alignements possibles

$A[0, i-1]$		$A[0, i]$		$A[0, i-1]$	
\dots		\dots		\dots	
\dots	$A[i]$	\dots	$B[j]$	\dots	$A[i]$
$B[0, j]$	-	$B[0, j-1]$		$B[0, j-1]$	$B[j]$

Alignements AGORR et ALGOR

AGORR	-	AGOR	R	AGOR	R	
ALGO	R	ALGOR	-	ALGO	R	
3	1	1	1	2	0	$\rightarrow 2$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve : Notation $\left| \begin{smallmatrix} A[0, i] \\ B[0, j] \end{smallmatrix} \right|$: alignement, avec $\text{edit}(i, j)$ désaccords

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve : Notation $\left| \begin{smallmatrix} A[0,i] \\ B[0,j] \end{smallmatrix} \right|$: alignement, avec $\text{edit}(i, j)$ désaccords

► $\text{edit}(i, j) \leq \text{edit}(i-1, j) + 1 : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j] \end{smallmatrix} \right| + \text{suppression } A[i]$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve : Notation $\left| \begin{smallmatrix} A[0,i] \\ B[0,j] \end{smallmatrix} \right|$: alignement, avec $\text{edit}(i, j)$ désaccords

- $\text{edit}(i, j) \leq \text{edit}(i-1, j) + 1 : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j] \end{smallmatrix} \right| + \text{suppression } A[i]$
- $\text{edit}(i, j) \leq \text{edit}(i, j-1) + 1 : \left| \begin{smallmatrix} A[0,i] \\ B[0,j-1] \end{smallmatrix} \right| + \text{insertion } B[j]$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve : Notation $\left| \begin{smallmatrix} A[0,i] \\ B[0,j] \end{smallmatrix} \right|$: alignement, avec $\text{edit}(i, j)$ désaccords

- ▶ $\text{edit}(i, j) \leq \text{edit}(i-1, j) + 1 : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j] \end{smallmatrix} \right| + \text{suppression } A[i]$
- ▶ $\text{edit}(i, j) \leq \text{edit}(i, j-1) + 1 : \left| \begin{smallmatrix} A[0,i] \\ B[0,j-1] \end{smallmatrix} \right| + \text{insertion } B[j]$
- ▶ $\text{edit}(i, j) \leq \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j-1] \end{smallmatrix} \right| + \text{remplacement } A[i] \rightarrow B[j]$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve : Notation $\left| \begin{smallmatrix} A[0,i] \\ B[0,j] \end{smallmatrix} \right|$: alignement, avec $\text{edit}(i, j)$ désaccords

- ▶ $\text{edit}(i, j) \leq \text{edit}(i-1, j) + 1 : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j] \end{smallmatrix} \right| + \text{suppression } A[i]$
- ▶ $\text{edit}(i, j) \leq \text{edit}(i, j-1) + 1 : \left| \begin{smallmatrix} A[0,i] \\ B[0,j-1] \end{smallmatrix} \right| + \text{insertion } B[j]$
- ▶ $\text{edit}(i, j) \leq \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} : \left| \begin{smallmatrix} A[0,i-1] \\ B[0,j-1] \end{smallmatrix} \right| + \text{remplacement } A[i] \rightarrow B[j]$

$$\implies \text{edit}(i, j) \leq \min\{\dots\}$$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve Si $\text{edit}(i, j) = d$,

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve Si $\text{edit}(i, j) = d$,

$$\blacktriangleright A[i] = B[j] \implies \text{edit}(i-1, j-1) = d$$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve Si $\text{edit}(i, j) = d$,

- ▶ $A[i] = B[j] \implies \text{edit}(i-1, j-1) = d$
- ▶ $A[i] \neq B[j]$:
 - ▶ $\text{edit}(i-1, j-1) \geq d-1$
 - ▶ $\text{edit}(i, j-1) \geq d-1$
 - ▶ $\text{edit}(i-1, j) \geq d-1$

Formule récursive

Lemme

$$\text{edit}(i, j) = \min \begin{cases} \text{edit}(i-1, j) + 1 \\ \text{edit}(i, j-1) + 1 \\ \text{edit}(i-1, j-1) + 1_{A[i] \neq B[j]} \end{cases}$$

où $1_{A[i] \neq B[j]} = 1$ si $A[i] \neq B[j]$, 0 sinon

Preuve Si $\text{edit}(i, j) = d$,

- ▶ $A[i] = B[j] \implies \text{edit}(i-1, j-1) = d$
- ▶ $A[i] \neq B[j]$:
 - ▶ $\text{edit}(i-1, j-1) \geq d-1$
 - ▶ $\text{edit}(i, j-1) \geq d-1$
 - ▶ $\text{edit}(i-1, j) \geq d-1$

$$\implies \text{edit}(i, j) \geq \min\{\dots\}$$

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1									
L	2									
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0								
L	2									
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2									
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1								
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1							
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3									
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3		2							
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1							
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4									
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5									
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6									
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7									
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8									
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9									
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10									

Algorithme : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

L'algorithme

Algorithme : DISTANCEEDITION(A, B)

$(m, n) \leftarrow$ tailles de A et B

$E \leftarrow$ tableau de dimensions m par n

pour $i = 0$ à $m - 1$ faire

 pour $j = 0$ à $n - 1$ faire

$\epsilon \leftarrow 0$ si $A[i] = B[j]$, 1 sinon

$E[i, j] \leftarrow \min(E[i-1, j]+1, E[i, j-1]+1, E[i-1, j-1]+\epsilon)$

retourner $E[m - 1, n - 1]$

L'algorithme

Algorithme : DISTANCEEDITION(A, B)

$(m, n) \leftarrow$ tailles de A et B

$E \leftarrow$ tableau de dimensions m par n

pour $i = 0$ à $m - 1$ faire

 pour $j = 0$ à $n - 1$ faire

$\epsilon \leftarrow 0$ si $A[i] = B[j]$, 1 sinon

$E[i, j] \leftarrow \min(E[i-1, j]+1, E[i, j-1]+1, E[i-1, j-1]+\epsilon)$

retourner $E[m - 1, n - 1]$

► Convention : $E[-1, j] = j + 1$; $E[i, -1] = i + 1$

L'algorithme

Algorithme : DISTANCEEDITION(A, B)

$(m, n) \leftarrow$ tailles de A et B

$E \leftarrow$ tableau de dimensions m par n

pour $i = 0$ à $m - 1$ faire

 pour $j = 0$ à $n - 1$ faire

$\epsilon \leftarrow 0$ si $A[i] = B[j]$, 1 sinon

$E[i, j] \leftarrow \min(E[i-1, j]+1, E[i, j-1]+1, E[i-1, j-1]+\epsilon)$

retourner $E[m - 1, n - 1]$

► Convention : $E[-1, j] = j + 1$; $E[i, -1] = i + 1$

Lemme

L'algorithme DISTANCEEDITION renvoie $\delta(A, B)$ en temps $O(mn)$.

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	4	5	6
H	8	7	6	5	4	4	4	4	5	6
M	9	8	7	6	5	5	5	5	5	6
E	10	9	8	7	6	6	6	6	6	6

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	6	7	8
G	3	2	1	2	3	4	5	6	7	8
O	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

Reconstruction : exemple

		A	G	O	R	R	Y	T	N	E
A	0	1	2	3	4	5	6	7	8	9
L	1	0	1	2	3	4	5	6	7	8
G	2	1	1	2	3	4	5	6	7	8
O	3	2	1	2	3	4	5	6	7	8
R	4	3	2	1	2	3	4	5	6	7
R	5	4	3	2	1	2	3	4	5	6
I	6	5	4	3	2	2	3	4	5	6
T	7	6	5	4	3	3	3	3	4	5
H	8	7	6	5	4	4	4	4	4	5
M	9	8	7	6	5	5	5	5	5	5
E	10	9	8	7	6	6	6	6	6	5

AGORRYTNE \rightarrow ALGORRYTNE \rightarrow ALGORYTNE

ALGORITHM_E ← ALGORITHM_H ← ALGORITHM_I

Algorithme de reconstruction

Algorithme : ALIGNEMENT(A, B, \mathbf{E})

$(i, j) \leftarrow (m - 1, n - 1)$

tant que $i \geq 0$ et $j \geq 0$ faire

 si $E[i, j] = E[i - 1, j - 1]$ et $A[i] = B[j]$ alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j - 1] + 1$ alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j] + 1$ alors

 Insérer « _ » après $B[j]$; $i \leftarrow i - 1$

 sinon si $E[i, j] = E[i, j - 1] + 1$ alors

 Insérer « _ » après $A[i]$; $j \leftarrow j - 1$

tant que $i \geq 0$ faire Insérer « _ » en tête de B ; $i \leftarrow i - 1$

tant que $j \geq 0$ faire Insérer « _ » en tête de A ; $j \leftarrow j - 1$

retourner A et B

Correction et complexité

Lemme

L'algorithme ALIGNEMENT aligne les mots A et B avec $\delta(A, B)$ désaccords, en temps $O(m + n)$.

Algorithme : ALIGNEMENT(A, B, \mathbf{E})

$(i, j) \leftarrow (m - 1, n - 1)$

tant que $i \geq 0$ et $j \geq 0$ faire

 si $E[i, j] = E[i - 1, j - 1]$ et $A[i] = B[j]$

 alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j - 1] + 1$ alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j] + 1$ alors

 Insérer « - » après $B[j]$; $i \leftarrow i - 1$

 sinon si $E[i, j] = E[i, j - 1] + 1$ alors

 Insérer « - » après $A[i]$; $j \leftarrow j - 1$

tant que $i \geq 0$ faire

 Insérer « - » en tête de B ; $i \leftarrow i - 1$

tant que $j \geq 0$ faire

 Insérer « - » en tête de A ; $j \leftarrow j - 1$

retourner A et B

Correction et complexité

Lemme

L'algorithme ALIGNEMENT aligne les mots A et B avec $\delta(A, B)$ désaccords, en temps $O(m + n)$.

Algorithme : ALIGNEMENT(A, B, \mathbf{E})

$(i, j) \leftarrow (m - 1, n - 1)$

tant que $i \geq 0$ et $j \geq 0$ faire

 si $E[i, j] = E[i - 1, j - 1]$ et $A[i] = B[j]$

 alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j - 1] + 1$ alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j] + 1$ alors

 Insérer « - » après $B[j]$; $i \leftarrow i - 1$

 sinon si $E[i, j] = E[i, j - 1] + 1$ alors

 Insérer « - » après $A[i]$; $j \leftarrow j - 1$

tant que $i \geq 0$ faire

 Insérer « - » en tête de B ; $i \leftarrow i - 1$

tant que $j \geq 0$ faire

 Insérer « - » en tête de A ; $j \leftarrow j - 1$

retourner A et B

Preuve de complexité : à chaque tour, $i + j$ diminue de ≥ 1

Correction et complexité

Lemme

L'algorithme ALIGNEMENT aligne les mots A et B avec $\delta(A, B)$ désaccords, en temps $O(m + n)$.

Algorithme : ALIGNEMENT(A, B, E)

$(i, j) \leftarrow (m - 1, n - 1)$

tant que $i \geq 0$ et $j \geq 0$ faire

 si $E[i, j] = E[i - 1, j - 1]$ et $A[i] = B[j]$

 alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j - 1] + 1$ alors

$(i, j) \leftarrow (i - 1, j - 1)$

 sinon si $E[i, j] = E[i - 1, j] + 1$ alors

 Insérer « - » après $B[j]$; $i \leftarrow i - 1$

 sinon si $E[i, j] = E[i, j - 1] + 1$ alors

 Insérer « - » après $A[i]$; $j \leftarrow j - 1$

tant que $i \geq 0$ faire

 Insérer « - » en tête de B ; $i \leftarrow i - 1$

tant que $j \geq 0$ faire

 Insérer « - » en tête de A ; $j \leftarrow j - 1$

retourner A et B

Preuve de complexité : à chaque tour, $i + j$ diminue de ≥ 1

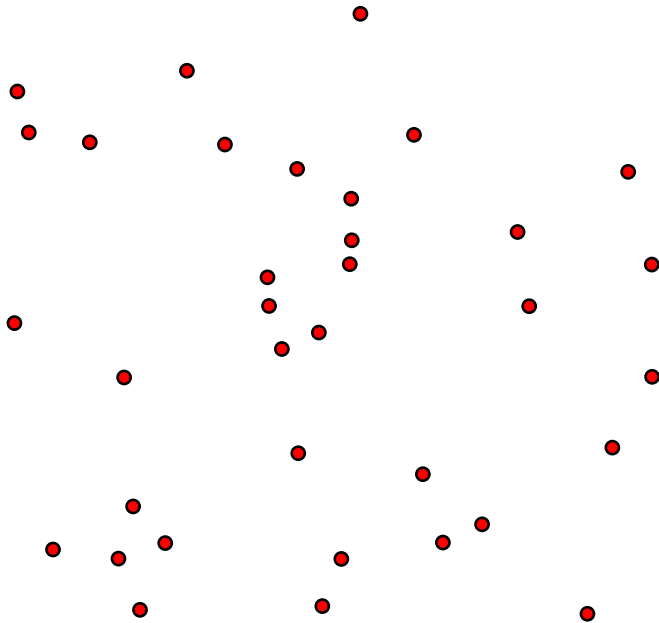
Preuve de correction : « en entrant dans la boucle,

$A[i + 1, m - 1]$ et $B[j + 1, n - 1]$ sont *alignés* avec

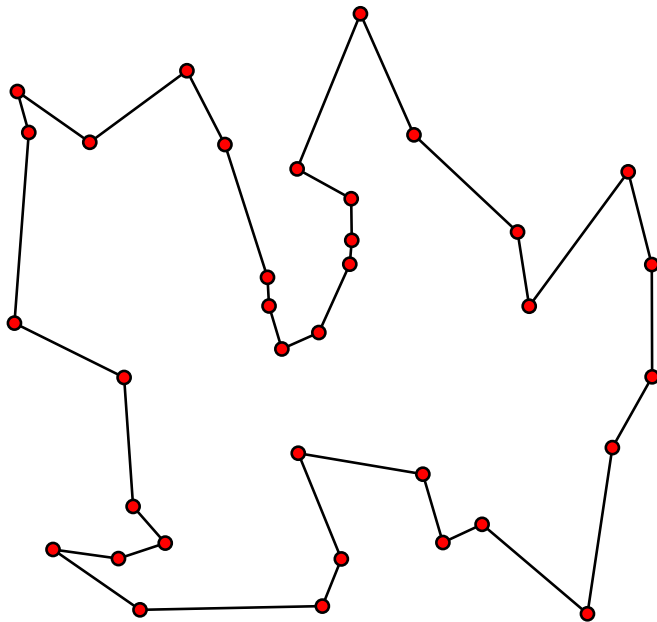
$\delta(A[i + 1, m - 1], B[j + 1, n - 1])$ désaccords »

1. Premier exemple : plus longue sous-suite croissante
2. Qu'est-ce que la programmation dynamique ?
3. Deuxième exemple : choix de cours, le retour
4. Troisième exemple : la distance d'édition
5. Exemple spécial : le voyageur de commerce

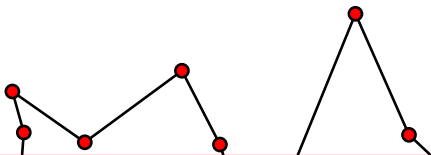
Chemin le plus court passant par chaque point ?



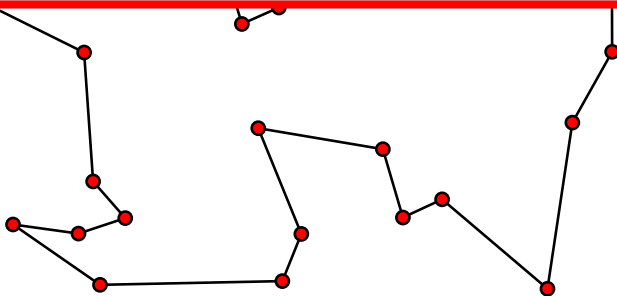
Chemin le plus court passant par chaque point ?



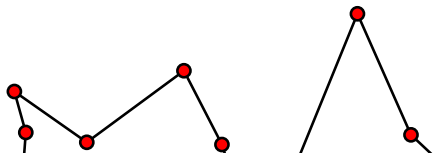
Chemin le plus court passant par chaque point ?

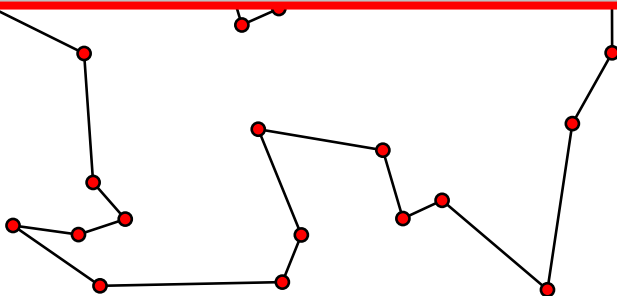


- ▶ Algorithme naïf : essayer toutes les permutations $\rightsquigarrow O(n!)$

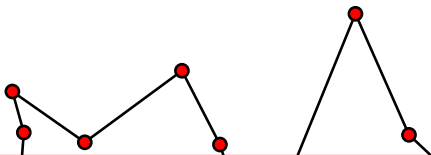


Chemin le plus court passant par chaque point ?

- 
- ▶ Algorithme naïf : essayer toutes les permutations $\rightsquigarrow O(n!)$
 - ▶ Algorithme polynomial : impossible sauf si $P = NP$

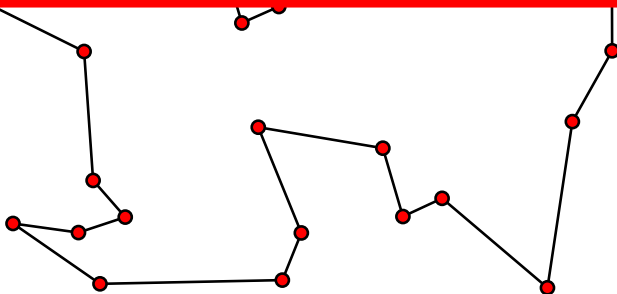


Chemin le plus court passant par chaque point ?



- ▶ Algorithme naïf : essayer toutes les permutations $\rightsquigarrow O(n!)$
- ▶ Algorithme polynomial : impossible sauf si $P = NP$

Objectif : algorithme mieux que $O(n!)$, mais pas polynomial



Formalisation

Entrée Ensemble $S = \{s_0, \dots, s_{n-1}\}$ de points dans le plan

Sortie 1 Chemin $s_{i_0} \rightarrow s_{i_1} \rightarrow \dots \rightarrow s_{i_{n-1}} \rightarrow s_{i_0}$ qui minimise la distance totale, avec $\delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ si $s_i = (x_i, y_i)$ et $s_j = (x_j, y_j)$

Sortie 2 Longueur $\ell_S = \sum_{j=0}^{n-1} \delta_{i_j i_{j+1}}$ du chemin le plus court ($i_n = i_0$)

Formalisation

Entrée Ensemble $S = \{s_0, \dots, s_{n-1}\}$ de points dans le plan

Sortie 1 Chemin $s_{i_0} \rightarrow s_{i_1} \rightarrow \dots \rightarrow s_{i_{n-1}} \rightarrow s_{i_0}$ qui minimise la distance totale, avec $\delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ si $s_i = (x_i, y_i)$ et $s_j = (x_j, y_j)$

Sortie 2 Longueur $\ell_S = \sum_{j=0}^{n-1} \delta_{i_j i_{j+1}}$ du chemin le plus court ($i_n = i_0$)

► On peut fixer $s_{i_0} = s_0$

Formalisation

Entrée Ensemble $S = \{s_0, \dots, s_{n-1}\}$ de points dans le plan

Sortie 1 Chemin $s_{i_0} \rightarrow s_{i_1} \rightarrow \dots \rightarrow s_{i_{n-1}} \rightarrow s_{i_0}$ qui minimise la distance totale, avec $\delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ si $s_i = (x_i, y_i)$ et $s_j = (x_j, y_j)$

Sortie 2 Longueur $\ell_S = \sum_{j=0}^{n-1} \delta_{i_j i_{j+1}}$ du chemin le plus court ($i_n = i_0$)

- ▶ On peut fixer $s_{i_0} = s_0$
- ▶ Si $U \subset S$ avec $s_0, s_j \in U$, on note $\Delta(U, s_j)$ la longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

Formalisation

Entrée Ensemble $S = \{s_0, \dots, s_{n-1}\}$ de points dans le plan

Sortie 1 Chemin $s_{i_0} \rightarrow s_{i_1} \rightarrow \dots \rightarrow s_{i_{n-1}} \rightarrow s_{i_0}$ qui minimise la distance totale, avec $\delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ si $s_i = (x_i, y_i)$ et $s_j = (x_j, y_j)$

Sortie 2 Longueur $\ell_S = \sum_{j=0}^{n-1} \delta_{i_j i_{j+1}}$ du chemin le plus court ($i_n = i_0$)

- ▶ On peut fixer $s_{i_0} = s_0$
- ▶ Si $U \subset S$ avec $s_0, s_j \in U$, on note $\Delta(U, s_j)$ la longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

$$\ell_S = \min_j \Delta(\{s_0, \dots, s_{n-1}\}, s_j) + \delta_{j,0}$$

Formule récursive pour Δ

$\Delta(U, s_j)$: longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

Formule récursive pour Δ

$\Delta(U, s_j)$: longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

Lemme

- ▶ $\Delta(\{s_0\}, s_0) = 0$ et $\Delta(U, s_0) = +\infty$ si $|U| > 1$
- ▶ $\Delta(U, s_j) = \min_{i \in U: i \neq j} \Delta(U \setminus \{s_j\}, s_i) + \delta_{ij}$

Formule récursive pour Δ

$\Delta(U, s_j)$: longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

Lemme

- ▶ $\Delta(\{s_0\}, s_0) = 0$ et $\Delta(U, s_0) = +\infty$ si $|U| > 1$
- ▶ $\Delta(U, s_j) = \min_{i \in U: i \neq j} \Delta(U \setminus \{s_j\}, s_i) + \delta_{ij}$

Preuve : pour aller de s_0 à s_j ,

- ▶ on choisit un point s_i
- ▶ on va de s_0 à s_i
- ▶ puis directement de s_i à s_j

Formule récursive pour Δ

$\Delta(U, s_j)$: longueur du plus court chemin de s_0 à s_j visitant chaque $s_i \in U$ une fois exactement

Lemme

- ▶ $\Delta(\{s_0\}, s_0) = 0$ et $\Delta(U, s_0) = +\infty$ si $|U| > 1$
- ▶ $\Delta(U, s_j) = \min_{i \in U: i \neq j} \Delta(U \setminus \{s_j\}, s_i) + \delta_{ij}$

Preuve : pour aller de s_0 à s_j ,

- ▶ on choisit un point s_i
- ▶ on va de s_0 à s_i
- ▶ puis directement de s_i à s_j
 \rightsquigarrow il *suffit* de trouver le meilleur s_i !

Algorithme TSP

Algorithme : TSP(S)

$\Delta \leftarrow$ tableau à 2 dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n - 1$

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour *tous les* $U \subset S$ de taille s tels que $s_0 \in U$ faire

$\Delta[U, s_0] = +\infty$

 pour *tout* $s_j \in U, j \neq 0$ faire

$\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

retourner $\min_j(\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$

Algorithme TSP

Algorithme : TSP(S)

$\Delta \leftarrow$ tableau à 2 dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n - 1$

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour *tous les* $U \subset S$ de taille s tels que $s_0 \in U$ faire

$\Delta[U, s_0] = +\infty$

 pour *tout* $s_j \in U, j \neq 0$ faire

$\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

retourner $\min_j (\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$

Lemme

L'algorithme TSP calcule ℓ_S en temps $O(n^2 2^n)$

Preuve de la complexité

Algorithme : TSP(S)

$\Delta \leftarrow$ tableau à deux dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n-1$

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour tous les $U \subset S$ de taille s tels que $s_0 \in U$ faire
 $\Delta[U, s_0] = +\infty$
 pour toute $s_j \in U, j \neq 0$ faire
 $\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

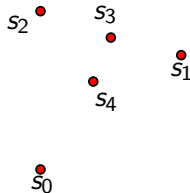
retourner $\min_j (\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$

Preuve

- Calcul du min : $O(s)$ car $|U| = s$
- Boucle sur s_j : $O(s^2)$
- Boucle sur U : $\binom{n-1}{s-1} O(s^2)$ car $\binom{n-1}{s-1}$ ss-ens.
- Boucle sur s : $\sum_{s=2}^n \binom{n-1}{s-1} O(s^2) \leq O(n^2) \sum_s \binom{n}{s} = O(n^2 2^n)$

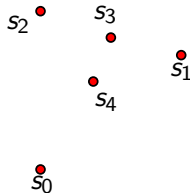
Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$				
$\{s_0, s_2\}$	$+\infty$				
$\{s_0, s_3\}$	$+\infty$				
$\{s_0, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2\}$	$+\infty$				
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



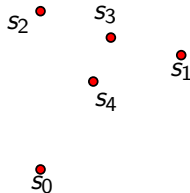
Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$				
$\{s_0, s_3\}$	$+\infty$				
$\{s_0, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2\}$	$+\infty$				
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$				
$\{s_0, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2\}$	$+\infty$				
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2\}$	$+\infty$				
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$				
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7			
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$				
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$				
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					



Exemple



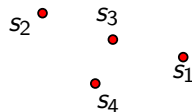
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$				
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



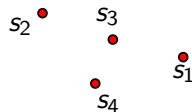
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



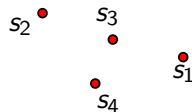
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$				
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$				
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$				
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$				
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



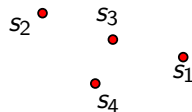
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$				
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$					

Exemple



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple



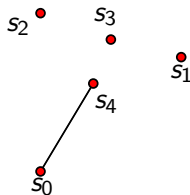
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



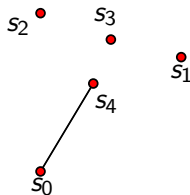
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



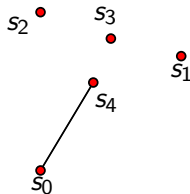
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



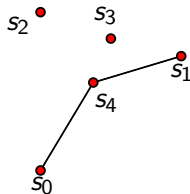
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



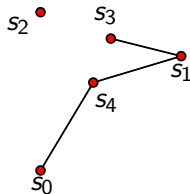
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



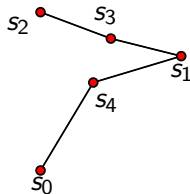
	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

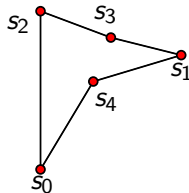
Exemple : reconstruction



	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9

Exemple : reconstruction

	s_0	s_1	s_2	s_3	s_4
$\{s_0\}$	0				
$\{s_0, s_1\}$	$+\infty$	20, 6			
$\{s_0, s_2\}$	$+\infty$		18		
$\{s_0, s_3\}$	$+\infty$			17	
$\{s_0, s_4\}$	$+\infty$				11, 7
$\{s_0, s_1, s_2\}$	$+\infty$	34, 7	37, 4		
$\{s_0, s_1, s_3\}$	$+\infty$	25, 2		28, 9	
$\{s_0, s_1, s_4\}$	$+\infty$	22, 1			31, 1
$\{s_0, s_2, s_3\}$	$+\infty$		25, 5	26, 5	
$\{s_0, s_2, s_4\}$	$+\infty$		21, 7		28
$\{s_0, s_3, s_4\}$	$+\infty$			17, 0	22, 4
$\{s_0, s_1, s_2, s_3\}$	$+\infty$	34, 8	37, 4	43, 0	
$\{s_0, s_1, s_2, s_4\}$	$+\infty$	38, 4	38, 9		45, 2
$\{s_0, s_1, s_3, s_4\}$	$+\infty$	25, 3		30, 3	34, 2
$\{s_0, s_2, s_3, s_4\}$	$+\infty$		25, 6	30, 2	31, 9
$\{s_0, s_1, s_2, s_3, s_4\}$	$+\infty$	38, 5	38, 9	46, 7	45, 2
$\min_j(\Delta[S, j] + \delta_{j0})$		59, 1	56, 9	63, 7	56, 9



Algorithme de reconstruction

Algorithme : $\text{TSP}(S)$

$\Delta \leftarrow$ tableau à deux dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n - 1$

$\text{Prec} \leftarrow$ tableau de mêmes dimensions

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour *tous les* $U \subset S$ de taille s tels que $s_0 \in U$ faire

$\Delta[U, s_0] = +\infty$

 pour *tout* $s_j \in U, j \neq 0$ faire

$\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

$\text{Prec}[U, s_j] \leftarrow$ indice du minimum

retourner $\min_j (\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$,

indice du min et Prec

Algorithme de reconstruction

Algorithme : $\text{TSP}(S)$

$\Delta \leftarrow$ tableau à deux dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n - 1$

$\text{Prec} \leftarrow$ tableau de mêmes dimensions

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour tous les $U \subset S$ de taille s tels que $s_0 \in U$ faire

$\Delta[U, s_0] = +\infty$

 pour tout $s_j \in U, j \neq 0$ faire

$\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

$\text{Prec}[U, s_j] \leftarrow$ indice du minimum

retourner $\min_j (\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$,

 indice du min et Prec

Algorithme :

$\text{TSP-REC}(S, \Delta, \text{Prec}, j)$

$i_0 \leftarrow 0$

$i_1 \leftarrow j$

$U \leftarrow S$

pour $k = 2$ à $n - 1$ faire

$i_k \leftarrow \text{Prec}[U, i_{k-1}]$

$U \leftarrow S \setminus \{s_{i_{k-1}}\}$

retourner

$(i_0, i_1, \dots, i_{n-1}, i_0)$

Algorithme de reconstruction

Algorithme : $\text{TSP}(S)$

$\Delta \leftarrow$ tableau à deux dimensions, indexé par les sous-ensembles de S contenant $\{s_0\}$, et par les entiers de 0 à $n-1$

$\text{Prec} \leftarrow$ tableau de mêmes dimensions

$\Delta[\{s_0\}, 0] = 0$

pour $s = 2$ à n faire

 pour tous les $U \subset S$ de taille s tels que $s_0 \in U$ faire

$\Delta[U, s_0] = +\infty$

 pour tout $s_j \in U, j \neq 0$ faire

$\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$

$\text{Prec}[U, s_j] \leftarrow$ indice du minimum

retourner $\min_j (\Delta[\{s_0, \dots, s_n\}, s_j] + \delta_{j0})$,

 indice du min et Prec

Algorithme :

$\text{TSP-REC}(S, \Delta, \text{Prec}, j)$

$i_0 \leftarrow 0$

$i_1 \leftarrow j$

$U \leftarrow S$

pour $k = 2$ à $n-1$ faire

$i_k \leftarrow \text{Prec}[U, i_{k-1}]$

$U \leftarrow S \setminus \{s_{i_{k-1}}\}$

retourner

$(i_0, i_1, \dots, i_{n-1}, i_0)$

Lemme

L'algorithme TSP-REC construit un chemin de longueur minimale en temps $O(n)$

Conclusion

Théorème

La longueur minimale d'un chemin passant par n points peut être calculée en temps $O(n^2 2^n)$. Le chemin lui-même peut être calculé en temps $O(n)$ supplémentaire.

Conclusion

Théorème

La longueur minimale d'un chemin passant par n points peut être calculée en temps $O(n^2 2^n)$. Le chemin lui-même peut être calculé en temps $O(n)$ supplémentaire.

- ▶ Détails à régler : gestion des ensembles
 - ▶ théorie : pas de problème (bonne complexité)
 - ▶ pratique : pas si facile !

Conclusion

Théorème

La longueur minimale d'un chemin passant par n points peut être calculée en temps $O(n^2 2^n)$. Le chemin lui-même peut être calculé en temps $O(n)$ supplémentaire.

- ▶ Détails à régler : gestion des ensembles
 - ▶ théorie : pas de problème (bonne complexité)
 - ▶ pratique : pas si facile !
- ▶ Fonctionne aussi hors d'un plan euclidien (carte, plans, graphes, ...)

Conclusion

Théorème

La longueur minimale d'un chemin passant par n points peut être calculée en temps $O(n^2 2^n)$. Le chemin lui-même peut être calculé en temps $O(n)$ supplémentaire.

- ▶ Détails à régler : gestion des ensembles
 - ▶ théorie : pas de problème (bonne complexité)
 - ▶ pratique : pas si facile !
- ▶ Fonctionne aussi hors d'un plan euclidien (carte, plans, graphes, ...)
- ▶ Très utile en pratique comme en théorie !

Conclusion

Théorème

La longueur minimale d'un chemin passant par n points peut être calculée en temps $O(n^2 2^n)$. Le chemin lui-même peut être calculé en temps $O(n)$ supplémentaire.

- ▶ Détails à régler : gestion des ensembles
 - ▶ théorie : pas de problème (bonne complexité)
 - ▶ pratique : pas si facile !
- ▶ Fonctionne aussi hors d'un plan euclidien (carte, plans, graphes, ...)
- ▶ Très utile en pratique comme en théorie !
- ▶ Un exemple : <http://map.vroom-project.org/>