

**IFT436 - Algorithmes et structure de données Devoir 5**

VALL-VILLELLAS Virgile (21 172 678)

LAPOINTE William (17 011 458)

*Table des matières*

[Question 1 : 3](#_Toc88672310)

[a) Si f(h) = h, alors on peut calculer la durée d’une descente minimale en temps O(1). Pourquoi? Justifiez. 3](#_Toc88672311)

[b) Donnez un algorithme de force brute qui identifie la durée d’une descente minimale en explorant toutes les descentes. Votre algorithme fonctionne-t-il en temps polynomial par rapport à n? Justifiez brièvement. 3](#_Toc88672312)

[c) Donnez un algorithme qui exploite la programmation dynamique avec tableaux afin d’identifier la durée d’une descente minimale. Votre algorithme doit fonctionner en temps O(n^2). Quel tableau est calculé par l’algorithme sur la montagne ci-dessus avec ? Vous pouvez donner les valeurs symboliquement ou numériquement arrondies à cinq décimaux après la virgule. 3](#_Toc88672313)

[Question 2 : 4](#_Toc88672314)

[a) question. 4](#_Toc88672315)

# Question 1 :

## a) Si f(h) = h, alors on peut calculer la durée d’une descente minimale en temps O(1). Pourquoi? Justifiez.

Nous souhaitons prouver que l’on peut calculer la durée d’une descente minimale en O(1) lorsque f(h) = f, nous allons démonter cela grâce à un exemple.

On sait que

Donc

Supposons la matrice suivante :

Reviens à écrire

En prenant n’importe quel chemin, a chaque fois on soustrait une position b a une position a, puis on soustrait une position d a une position b et ainsi de suite jusqu’à arriver à la fin. On remarque donc que le seul sommet que nous n’additionnons pas est le dernier (ici d) puisqu’on ne peut pas descendre en dessous. Et le seul sommet qu’on ne soustrait pas est le premier de la montagne puisque on ne descend jamais vers lui (a).

On en conclut donc que pour calculer la durée d’une descente minimale en temps O(1), il suffit de soustraire le dernier sommet au premier.

Distance minimale = M[0][0] – M[n][n]

## b) Donnez un algorithme de force brute qui identifie la durée d’une descente minimale en explorant toutes les descentes. Votre algorithme fonctionne-t-il en temps polynomial par rapport à n? Justifiez brièvement.

Spécifications :

* Entrée : Matrice d’entier : tab de taille n\*n
* Pré-condition :
* Sortie :
* Post condition :

meilleurChemin(Matrice d’entier : tab)

aux(entier : sol, entier : i, entier : j)

**Si**

**retourner** sol

**SinonSi**

**retourner** *aux*(sol + *sqrt*(tab[i][j] - tab[i+1][j]), i+1, j)

**SinonSi**

**retourner** *aux*(sol + *sqrt*(tab[i][j] - tab[i][j+1]), i, j+1)

**Sinon**:

**retourner** *min*(*aux*(sol + *sqrt*(tab[i][j] - tab[i+1][j]), i+1, j),

*aux*(sol + *sqrt*(tab[i][j] - tab[i][j+1]), i, j+1))

**retourner** *aux*(0, 0, 0)

entier : sol\_finale 🡨 0 ;

sol\_finale = *meilleurChemin*(tab)

## c) Donnez un algorithme qui exploite la programmation dynamique avec tableaux afin d’identifier la durée d’une descente minimale. Votre algorithme doit fonctionner en temps O(n^2). Quel tableau est calculé par l’algorithme sur la montagne ci-dessus avec ? Vous pouvez donner les valeurs symboliquement ou numériquement arrondies à cinq décimaux après la virgule.

Je pense qu’il faut améliorer l’algorithme de Djikstra

## Question 3 :

## a) Exécutez l’algorithme de Dijkstra sur le graphe pondéré suivant à partir du sommet a. Laissez une trace de chaque itération de l’algorithme en indiquant les sommets marqués et les distances partielles identifiées.

Rouge = visité

Orange = en cours de visite

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | infini |  |
| c | Infini |  |
| d | Infini |  |
| e | Infini |  |
| f | infini |  |

Une image contenant texte, horloge

Description générée automatiquement

Une image contenant texte, horloge

Description générée automatiquement

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 4 | a |
| d | Infini |  |
| e | Infini |  |
| f | infini |  |

On visite ensuite le sommet non marqué avec la plus petite distance par rapport au sommet ‘a’, dans notre cas le sommet ‘b’

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 4 ou 1+2 ? | a ou b ? |
| d | 1+4 | b |
| e | Infini |  |
| f | infini |  |

Une image contenant texte, horloge

Description générée automatiquement

On voit que b peut atteindre ‘c’ et propose une meilleur alternative de parcours pour atteindre ce sommet, donc on remplace la distance minimale de ‘a’ du sommet ‘c’ tout en précisant de quel sommet on vient.

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 3 | b |
| d | 5 ou 3+1 ? | b ou c ? |
| e | 3+3 | c |
| f | infini |  |

Une image contenant texte, horloge

Description générée automatiquement

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 3 | b |
| d | 4 | c |
| e | 6 | d |
| f | 6 | d |

Une image contenant texte, horloge

Description générée automatiquement

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 3 | b |
| d | 4 | c |
| e | 6 | d |
| f | 6 | d |

Une image contenant texte, horloge

Description générée automatiquement

|  |  |  |
| --- | --- | --- |
| Sommet | Distance minimale de ‘a’ | Sommet précédent |
| a | 0 |  |
| b | 1 | a |
| c | 3 | b |
| d | 4 | c |
| e | 6 | d |
| f | 6 | d |

Une image contenant texte, horloge

Description générée automatiquement

Tous les sommets ont été visités et sont marqués.