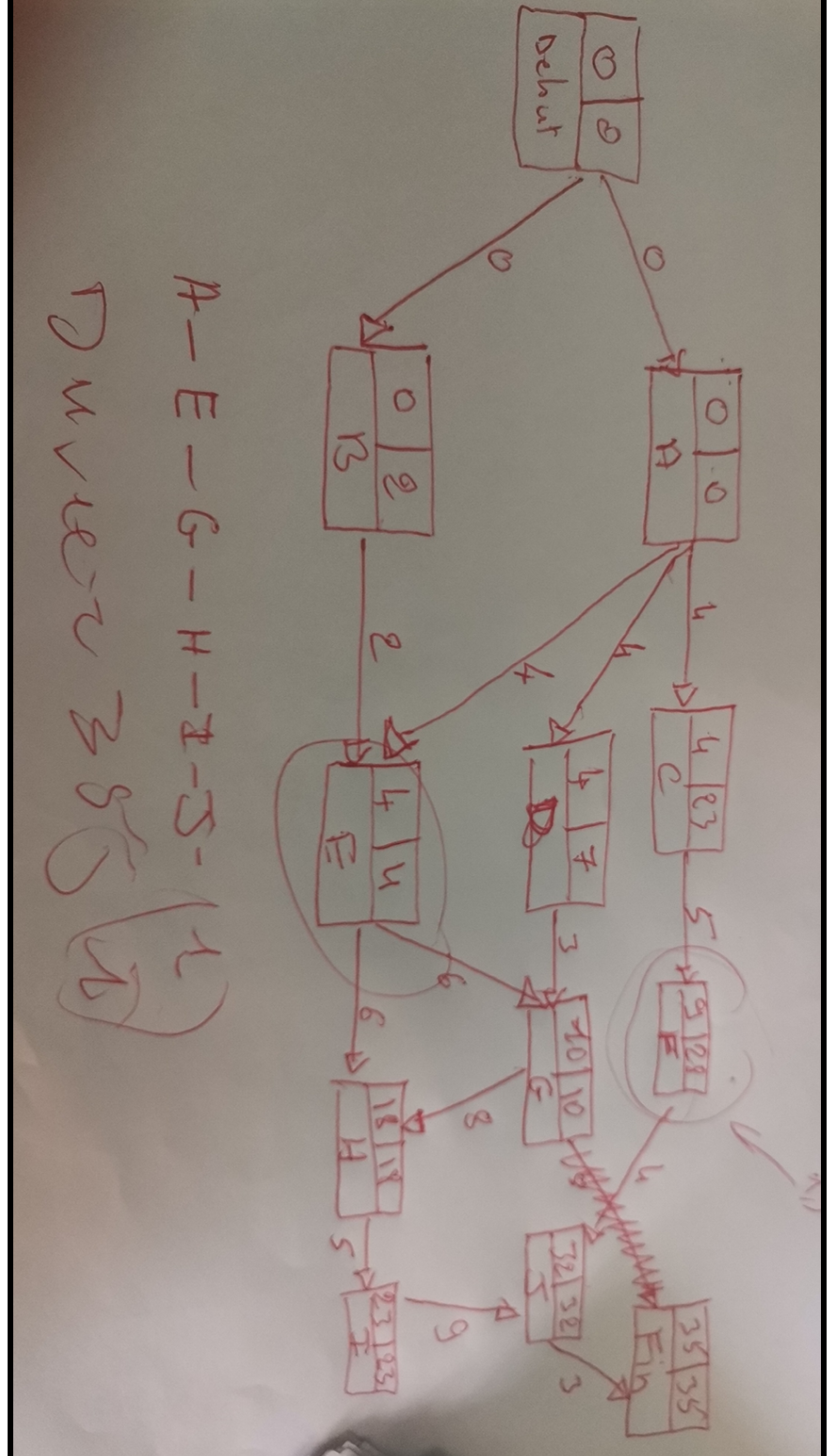
Exercice 1 : *(8 points)* La mise en service d’un nouvel équipement routier demande la réalisation d’un certain nombre de tâches. Le tableau ci-dessous représente ces différentes tâches avec leurs relations d’antériorité.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tâches | A | B | C | D | E | F | G | H | I | J |
| Durée | 4 | 2 | 5 | 3 | 6 | 4 | 8 | 5 | 9 | 3 |
| Tâches antérieurs | - | - | A | A | A,B | C | D,E | E,G | H | F,I |

1. Construire le graphe d’ordonnancement du projet avec la méthode des potentiels métra (MPM) et calculer les dates au plus tôt et au plus tard de chaque tache.
2. Déterminer le chemin critique. Quelle est la durée minimale de réalisation du projet ?
3. En réalité, la tâche C a nécessité une durée de 7 jours. Est-ce que cela a eu une incidence sur la durée de réalisation du projet ?
4. Calculer la marge totale de la tâche F. Qu’est-ce que cela signifie ?
5. Calculer la marge libre de E. Qu’est-ce que cela signifie ?

Réponse 1 :





**Réponse 2 :**

1. *Le chemin critique est : A – E – G – H – I -J*



1. *La durée du projet esr : 35*



**Réponse 3 :** *Même si la durée de la tache C est modifiée à 7, elle n’aura aucune incidence sur la durée de réalisation du projet.*



**Réponse 4 :**



1. *La marge totale de la tâche F est : 19*
2. *La marge totale est la durée que peut prendre une tache sans augmenter la durée totale du projet. En d’autres termes c’est une marge sur la tâche qui ne modifie pas la date de fin du projet.*



Réponse 5 :

1. *La marge libre de E est : 0.*



1. *La marge libre est le retard qui peut être autorisé sur une tache sans influencer la date de début au plutôt des taches qui lui succèdent.*



Exercice 2 : *(10 points)* On considère les algorithmes de recherche d’un élément dans un tableau :

|  |  |
| --- | --- |
| Recherche dichotomique  *Entrée : un tableau T de n entiers trié ; un entier x*  *Sortie : vrai ou faux selon que x est ou pas dans T*  *m, d, f : entier*  *trouve : boolean*  *d ← 0*  *f ← n*  *m un entier*  *trouve ← faux*  *tant que trouve égal faux et d inferieur à f faire*  *m ← (f - d)/2*  *si T[m] = x*  *trouve ← vrai;*  *sinon*  *si T[m] x*  *d ← m+1;*  *sinon*  *f ← m*  *fin tant que*  *retourne trouve;* | Recherche linéaire  *Entrées : un tableau T de n entiers trié ; un entier x*  *Sortie : vrai ou faux selon que x est ou pas dans T*  *i : entier*  *trouve : boolean*  *trouve ← faux*  *i ← 1*  *tant que ( trouve = faux et i n) faire*  *si T[i] = x*  *trouve ← vrai;*  *si t[i] x*  *retourne trouve*  *i ← i +1*  *fin tant que*  *retourne trouve;* |

1. **Calculer la complexité de l’algorithmes de Recherche linéaire**

|  |  |  |
| --- | --- | --- |
| *C1*  *C2*  *C3*  *C4* | *n+1 fois*  *n fois*  *n fois*  *n fois* | Recherche linéaire  *Entrées : un tableau T de n entiers trié ; un entier x*  *Sortie : vrai ou faux selon que x est ou pas dans T*  *i : entier*  *trouve : boolean*  *trouve ← faux*  *i ← 1*  *tant que ( trouve = faux et i n) faire*  *si T[i] = x*  *trouve ← vrai;*  *si t[i] x*  *retourne trouve*  *i ← i +1*  *fin tant que*  *retourne trouve;* |

*TLin = (n+1)C1+ n(C2+C3+C4) = n(C1+C2+C3+C4) +C1*



*En posant et , alors on a*

*Tlin=*



1. **Calculer la complexité de l’algorithmes de Recherche dichotomiqu.**

|  |  |  |
| --- | --- | --- |
| Recherche dichotomique  *Entrée : un tableau T de n entiers trié ; un entier x*  *Sortie : vrai ou faux selon que x est ou pas dans T*  *m, d, f : entier*  *trouve : boolean*  *d ← 0*  *f ← n*  *m un entier*  *trouve ← faux*  *tant que trouve égal faux et d inferieur à f faire*  *m ← (f - d)/2*  *si T[m] = x*  *trouve ← vrai;*  *sinon*  *si T[m] x*  *d ← m+1;*  *sinon*  *f ← m*  *fin tant que*  *retourne trouve;* | Nombre de foix    *log(n) +1*  *log(n)*  *log(n)*      log(n)  log(n) | *Nombre d’instructions élémentaires*  C1  C2  C3  C4  C5 |

TDico = C1 (*log(n) +1) + C(log(n)) + C3(log(n)) + C4(*log(n)) +C5(log(n) = log(n) (C1+C2+C3+C4+C5) + C1



*En posant et , alors on a*

TDico =  *=*



Réponse 2 : *L’algorithme de « Recherche dichotomique » est plus efficace, avec 5 itérations sur un tableau de 49 entiers, que celui de la « Recherche linéaire » qui termine au bout de 49 entier.*



Réponse 3 : A l’aide d’un invariant de boucle, prouver la validité de ces algorithmes

1. *« Recherche dichotomique » : l’invariant de boucle est « A la ième itération, les sous tableau T[0…d-1] et T[f+1…n] ne contient pas v. »*



*Initialisation : A la première itération, d=0 et f=n, les sous tableaux T[0…-1] est T[n+1…n] est vide et ne contient donc pas v.*



***Conservation :*** *A la ième itération, les sous tableaux T[0…d-1] et T[f+1…n] ne contiennent pas v. En calculant m=f-d et en constatant que T[m] ne contient pas v, alors nous avons deux choix possibles :*

* *T[m] , alors v n’est pas dans T[d…m]. Alors, en affectant m à la variable d, on a l’invariant pour l’itération suivante.*



* *T[m] , alors v n’est pas dans T[m…f]. Alors, en affectant m à la variable f, on a l’invariant pour l’itération suivante.*

***Terminaison :*** *A la terminaison, les sous tableaux T[0…d-1] et T[f+1…n] ne contiennent pas v. Si trouvé est à VRAI, alors c’est que v est dans T[m] et l’algorithme est correct. Sinon, on a dT[0…d-1] + T[f+1…n] et alors T ne contient pas v et l’algorithme est correct.*



1. *« Recherche linéaire » : L’invariant de boucle : « A la ième itération de boucle, le sous tableau T[1…i-1] ne contient pas v ».*



***Initialisation :*** *A la première itération, i =1. Aucun élément du tableau n’est traité.*



***Conservation :*** *A la ième itération de boucle, le sous tableau T[1…i-1] ne contient pas v et trouvé = FAUX. Si l’itération i modifie la valeur de trouvé à VRAI, alors l’itération i+1 ne serait pas fait et donc TROUVE reste à FAUX à la fin de l’itération i. T[i] ne contient donc pas v et l’invariant reste vérifié pour l’itération suivante.*



***Terminaison :***  *A la terminaison,*

*si i = n+1, alors le sous tableau T[1…i-1] = T[1…n]= T ne contient pas v et le programme est correcte en retournant FAUX.*



*Si i . Le programme est correct car i est la première case du tableau qui contient v.*

*Bonne chance !*