N° d'anonymat:

Année Universitaire 2024-2025
Session 1 de printemps

Parcours/Étape: L3 Info./MI/CMI **Code UE:** 4TIN602U
Épreuve: Techniques Algorithmiques et Programmation
Date: 22/04/2024 **Heure:** 9h00 **Durée:** 1h30
Documents: une seule feuille A4 recto-verso autorisée.
Épreuve de M. Cyril Gavoille

université de BORDEAUX
Collège Sciences et Technologie

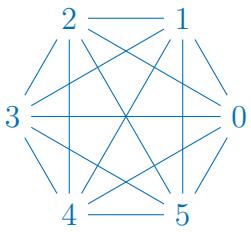
RÉPONDRE DIRECTEMENT SUR LE SUJET
QUI EST À RENDRE DANS LA FEUILLE DOUBLE D'EXAMEN

Question de cours

Dans l'algorithme ApproxMST, permettant d'approximer la solution optimale du Voyageur de commerce (V, d), nous avons été amenés à considérer le graphe complet à n sommets où les sommets correspondent aux n points de V, et les arêtes à toutes les distances entre les points de V.

Question 1. *Dessiner le graphe complet à $n = 6$ sommets et donnez son nombre d'arêtes.*

Solution. **[3 pts** $|\sum 3.0$**]** Il possède $5 + 4 + 3 + 2 + 1 = 10$ arêtes.



Question 2. *Donnez la formule générale donnant le nombre d'arêtes du graphe complet à n sommets. Justifiez.*

Solution. **[3 pts** $|\sum 6.0$**]** En épluchant les sommets et en sommant le degrés des sommets épluchés, cela donne : $(n-1) + (n-2) + \cdots + 3 + 2 + 1 = \sum_{i=1}^{n-1} i = n(n-1)/2$. C'est aussi le nombre de paires d'entiers $\{i, j\}$ pris dans $\{1, \ldots, n\}$, soit précisément $\binom{n}{2} = n(n-1)/2$.

Question 3. *Ordonnez de manière croissante les 9 complexités asymptotiques suivantes :*

$$2^n, \quad \ln(n)\ln\ln(n), \quad n, \quad e^n, \quad e^{n/\ln(n)}, \quad n^{(\ln(n))^2}, \quad n\ln(n), \quad n^2, \quad n\sqrt{n}$$

Vous pourrez écrire $f(n) < g(n)$ pour dire que la complexité $f(n)$ est asymptotiquement plus petite que la complexité $g(n)$.

Solution. **[5 pts** $|\sum 11.0$**]**

$$\ln(n)\ln\ln(n) \;<\; n \;<\; n\ln(n) \;<\; n\sqrt{n} \;<\; n^2 \;<\; n^{(\ln(n))^2} \;<\; e^{n/\ln(n)} \;<\; 2^n \;<\; e^n$$

Lorsqu'il y a des difficultés avec les exposants, comme $n^{(\ln(n))^2}$ vs. $e^{n/\ln(n)}$, un bon moyen pour les comparer est de comparer leurs logarithmes qui doivent être rangés dans le même ordre par

croissance de la fonction log. Ici cela donne $\ln\left(n^{\ln(n)}\right)$ vs. $\ln\left(e^{n/\ln(n)}\right)$, ce qui revient à comparer $(\ln(n))^2 \cdot \ln(n) = (\ln(n))^3$ à $n/\ln(n) \cdot \ln e = n/\ln(n)$, cette dernière étant clairement la plus grande.

J'ai noté en comptant le nombre de comparaisons adjacentes correctes. Par exemple, ... $n < 2^n < n^2 < e^n$ ... aurait contribué pour 2 points (sur 3 possibles). J'ai ensuite enlevé 3 points (une réponse aléatoire donnant 4 réponses justes en moyenne). Au final, cela donne une note entre 0 et 5.

Question 4. *Deux informaticiens ont chacun conçu un nouvel algorithme de tri. On s'intéresse à leur complexité asymptotique dans le pire des cas en nombre de comparaisons. Le premier affirme que son algorithme a une complexité bien précise de $f(n) = \lceil 2n\ln(n) \rceil$, alors que le deuxième annonce que son algorithme a une complexité $g(n) = O(n\ln(n))$. En terme de complexité, y a-t'il un algorithme meilleur que l'autre ? Si oui, lequel ? Discutez.*

SOLUTION. **[3 pts |∑14.0]** Aucune des complexités n'est meilleure que l'autre. En effet, $g(n) = O(n\ln n)$ signifie qu'ils existent deux constantes $n_0, c > 0$ telles que $\forall n > n_0$ (c'est-à-dire si n est assez grand), $g(n) \leqslant c \times n\ln n$. Donc suivant la valeur de c, on ne peut pas dire si, lorsque n est suffisamment grand, $f(n) < g(n)$ ou $f(n) > g(n)$.

Par exemple, on pourrait avoir $f(n) < g(n) = 10 \times n\ln(n)$ tout comme $f(n) > g(n) = 1.5 \times n\ln(n)$. La notation $g(n) = O(n\ln(n))$ n'exclut pas non plus le cas où $g(n) = \lceil \sqrt{n} \rceil$, même si ici cela n'est pas possible pour un algorithme de tri. Mais c'est une autre histoire.

Mine d'or

Un mineur souhaite extraire le meilleur filon d'une mine d'or. Une étude précise du terrain a permis de mesurer la quantité d'or disponible en kg dans chaque bloc de la mine, les blocs étant répartis en couches linéaires niveau par niveau. Le résultat est donné sous la forme d'une matrice M rectangulaire $H \times L$, où $M(i,j)$ indique la quantité d'or disponible à la profondeur $i \in \{0,\ldots,H-1\}$ et à la position $j \in \{0,\ldots,L-1\}$. On supposera $H, L \geqslant 1$.

Pour simplifier, on identifiera la mine avec sa matrice de mesures. Voici un exemple M_0 de mine 4×3.

$M_0 =$

$j=0$	$j=1$	$j=2$	
1	2	**5**	$i=0$
7	**5**	0	$i=1$
6	**7**	4	$i=2$
2	0	1	$i=3$

Le processus d'extraction d'un filon d'or dans M est le suivant. Le niveau $i = 0$ correspond à la surface de la mine et est imédiatement accessible au mineur. Le mineur est donc libre de choisir la position initiale comme il le souhaite. Une fois cette position j choisie, il creuse le bloc pour récupérer la quantité $M(i,j)$ d'or qui s'y trouve. De là, il peut accéder à l'un des blocs de niveau $i+1$ qui est soit juste en dessous, juste avant ou juste après la position j. Dit autrement, une fois le bloc (i,j) extrait, il peut extraire l'un des blocs $(i+1, j-1)$, $(i+1, j)$ ou $(i+1, j+1)$. Il continue ainsi de suite jusqu'à atteindre le fond de la mine. Un filon peut être vu comme un chemin de blocs reliant la première à la dernière ligne, qui ne fait que descendre avec des mouvements latéraux de -1, 0 ou +1.

Le but du mineur est d'extraire le meilleur filon de M, c'est-à-dire celui contenant la quantité maximum d'or. Pour M_0, le meilleur filon contient $5+5+7+2 = 19$ kg d'or et est composé des blocs $(0,2) \to (1,1) \to (2,1) \to (3,0)$.

Pour un bloc (i,j) d'une mine M donnée, on définit $f(i,j)$ comme la quantité d'or du meilleur filon à partir du bloc (i,j). On a vu pour M_0 que $f(0,2) = 19$. On a aussi $f(1,1) = 5+7+2 = 14$ et $f(2,2) = 4+1 = 5$.

Question 5. *Pour la mine M_0, déterminer $f(1,0)$, $f(0,0)$, $f(0,1)$, en précisant les calculs.*

SOLUTION. **[3 pts |∑17.0]** $f(1,0) = 7+7+2 = 16$, $f(0,0) = 1+7+7+2 = 17$, $f(0,1) = 2+7+7+2 = 18$.

Il est clair que le meilleur filon consiste à choisir une position initiale j telle que $f(0,j)$ est maximum. Autrement dit, la quantité d'or $Q(M)$ du meilleur filon vaut précisément

$$Q(M) \;=\; \max_{j\in\{0,\dots,L-1\}} f(0,j)$$

Grâce à la question 5, vous pouvez vérifier en effet que $Q(M_0) = \max\{f(0,0), f(0,1), f(0,2)\} = 19$.

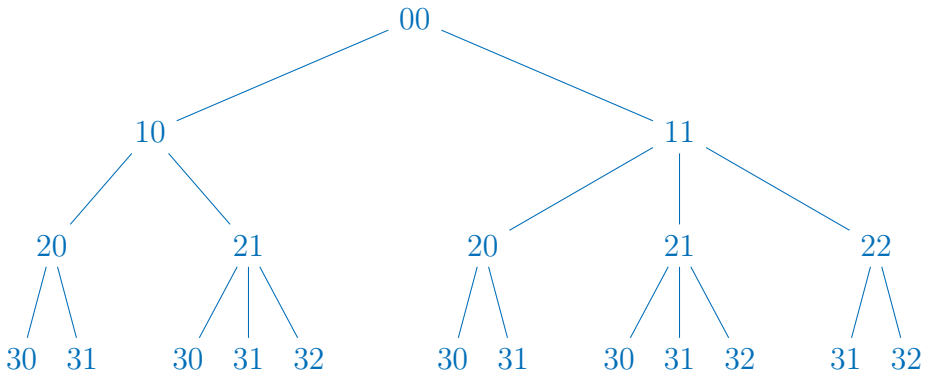
Question 6. *Pour une mine M quelconque, déterminez une formule de récurrence permettant de calculer $f(i,j)$. [Pour simplifier, posez $f(i,j) = 0$ si $i \notin \{0,\dots,H-1\}$ ou $j \notin \{0,\dots,L-1\}$.]*

SOLUTION. **[3 pts |∑20.0]**

$$f(i,j) = \begin{cases} 0 & \text{si } i \notin [0,H[\text{ ou } j \notin [0,L[\\ M(i,j) + \max\{f(i+1,j-1), f(i+1,j), f(i+1,j+1)\} & \text{si } i \in [0,H[\text{ et } j \in [0,L[\end{cases}$$

Question 7. *Déterminez le nombre de nœuds de l'arbre des appels de $f(0,0)$ pour la mine M_0 et dessinez l'arbre correspondant. Pour simplifiez, vous pourrez représenter l'appels (i,j) par "ij". [Aide : Représentez uniquement les appels valides et commencez par les (sous-)arbres de $f(1,0)$ et $f(1,1)$.]*

SOLUTION. **[3 pts |∑23.0]** Le nombre de noeuds pour $f(1,0)$ est 8 et pour $f(1,1)$ est 11. Pour $f(0,0)$, c'est donc $1+8+11 = 20$. Voici l'arbre.



Attention de ne pas confondre le nombre nœuds de l'arbre, c'est-à-dire le nombre de sommets, avec le nombre d'appels différents dans l'arbre, qui lui est moindre. Ce n'était pas demandé, mais dans l'arbre ci-dessus, il y avait 9 appels différents : 00, 10, 11, 20, 21, 22, 30, 31, 32.

Question 8. *De manière générale, en fonction de H et L, donnez un majorant pour le nombre de nœuds de l'arbre des appels de $f(0,j)$ pour une position j quelconque. Justifiez.*

SOLUTION. **[3 pts |∑26.0]** Le nombre de nœuds au niveau i est au plus 3^i, puisque chaque nœud a au plus 3 fils (cela peut être 2 pour les bords et même 1 seul si $L = 1$). Au total, il y a donc au plus $\sum_{i=0}^{H-1} 3^i = (3^H - 1)/2$. À cause des bords, il y en a moins de sommets, mais on demandait un majorant et une justification. NB : Si $L = 1$, il y en a que $O(H)$ nœuds.

Question 9. *En déduire la complexité du calcul de la quantité d'or du meilleur filon $Q(M)$ en utilisant le calcul récursif de $f(i,j)$.*

SOLUTION. **[3 pts |∑29.0]** D'après la question 8, c'est au plus $O(L \cdot 3^H)$, puisqu'on doit calculer $f(0,j)$ pour chaque $j \in \{0,\dots,L\}$.

Il est facile de voir qu'avec cette méthode récursive pour $Q(M)$ il existe des calculs inutiles. Par exemple, le calcul de $f(0,0)$ utilise $f(1,0)$ et $f(1,1)$, mais le calcul de $f(0,1)$ utilise aussi ces deux

valeurs. On se propose donc d'employer la programmation dynamique. Pour cela, on va utiliser un tableau `double F[H][L];` où `F[i][j]`$= f(i,j)$ tout simplement.

Question 10. *Donnez le tableau* `F` *rempli pour la mine* M_0. *[Aide : La dernière ligne est facile à construire et vous devriez avoir* `F[0][2]`$= 19$.*]*

SOLUTION. **[3 pts** $|\sum 32.0|$**]** La table complète est celle-ci, que l'on remplit facilement de la dernière ligne à la première, chaque case étant le maximun de deux ou trois cases juste en dessous :

$\texttt{F} =$

0	1	2	
17	18	19	0
16	14	9	1
8	9	5	2
2	0	1	3

Question 11. *En supposant que* `H` *et* `L` *sont des variables globales, écrivez la partie manquante du code ci-dessous censé calculer par programmation dynamique* $Q(M)$ *pour une mine* M *quelconque. [Aide : Faites attention aux bords de la mine et au cas* $L = 1$.*]*

```
int const H,L;
double Q(double M[H][L]){
  double F[H][L];
  ...; // partie à compléter dans le cadre ci-dessous
  int q=0;
  for(int j=0; j<L; j++) q = fmax(q, F[0][j]);
  return q;
}
```

SOLUTION. **[3 pts** $|\sum 35.0|$**]**

```
for(int j=0; j<L; j++) F[H-1][j] = M[i][j]; // dernière ligne
double pred,mid,succ; // bloc à gauche, en dessous, à droit
for(int i=H-2; i>=0; i--) // en montant depuis l'avant dernière ligne
  for(int j=0; j<L; j++){
    mid = pred = succ = F[i+1][j]; // bloc en dessous par défaut
    if(j>0)   pred = F[i+1][j-1];  // bloc à gauche existe
    if(j<L-1) succ = F[i+1][j+1];  // bloc à droit existe
    F[i][j] = M[i][j] + fmax(pred,fmax(mid,succ));
  }
```

Question 12. *Quelle est la complexité de* `Q(M)` *en fonction des dimensions de* `M` *? Justifiez.*

SOLUTION. **[3 pts** $|\sum 38.0|$**]** Sa complexité est $O(H \cdot L)$ à cause des deux boucles `for` imbriquées, la première et dernière boucle étant négligeables car en $O(L)$.

Seaux d'eau

On dispose d'une batterie de n seaux sans graduation. La capacité est variable suivant le seau mais est toujours un nombre entier de litres. Par exemple $(4, 3, 1)$ représente une batterie de $n = 3$ seaux de capacités 4, 3 et 1 litres respectivement.

Initialement, le premier seau de la batterie est rempli d'eau, tous les autres étant vides. Une configuration est une suite qui indique le nombre de litres d'eau contenu dans chacun des seaux. Donc, pour la batterie $(4, 3, 1)$, la configuration initiale est $(4, 0, 0)$.

L'objectif est de savoir s'il est possible, à l'aide d'opérations de transvasement, d'obtenir une configuration donnée. Par exemple, à partir de la configuration $S = (4, 0, 0)$ on peut obtenir la configuration $T = (2, 2, 0)$ en versant le 1er seau dans le 2e, ce qui produit la configuration $(1, 3, 0)$, puis en versant le 2e seau dans le 3e, ce qui produit $(1, 2, 1)$, et enfin en versant le 3e seau vers le 1er. Cela produit ainsi un chemin avec trois versements : $(4, 0, 0) \to (1, 3, 0) \to (1, 2, 1) \to (2, 2, 0)$. Bien entendu, les versements doivent s'effectuer complètement, sans perte d'eau et en respectant la capacité des seaux, c'est-à-dire que le transvasement s'arrête dès que le seau qu'on remplit est plein ou que le seau qu'on vide est vide.

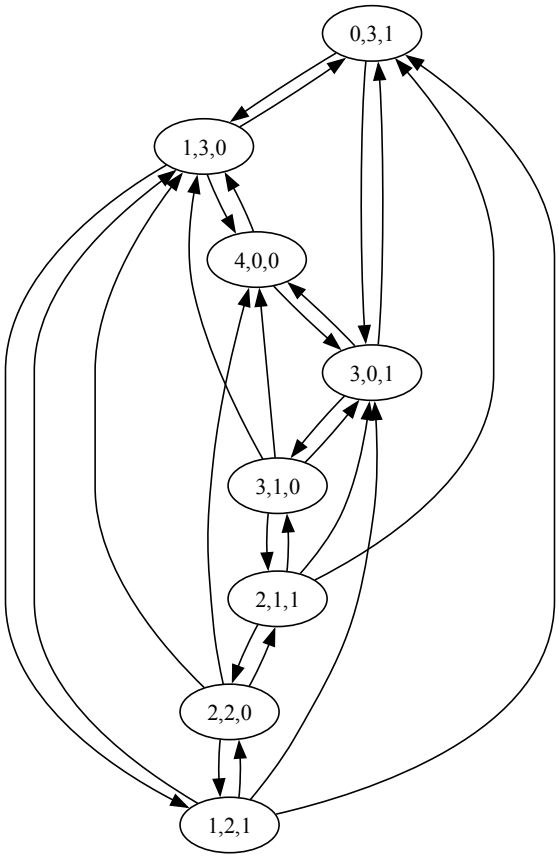
En `C` on aurait pu écrire :

```c
#define n 3
typedef int batterie[n];
typedef int config[n];
batterie B={4,3,1};
config S={4,0,0};
config T={2,2,0};
```

Étant donnée une batterie $B = (b_0, \cdots, b_{n-1})$ de n seaux, on définit le graphe G_B comme le graphe des configurations atteignables depuis la configuration initiale $S = (b_0, 0, \ldots, 0)$, deux configurations $X = (x_0, \ldots, x_{n-1})$ et $Y = (y_0, \ldots, y_{n-1})$ de G_B étant connectées par un arc $X \to Y$ si $Y \neq X$ et s'il existe un transvasement du seau i vers le seau j pour un certain couple $(i, j) \in \{0, \ldots, n-1\}^2$.

Question 13. *Dessinez le graphe G_B pour $B = (4, 3, 1)$. Le graphe est-il symétrique ? (c'est-à-dire, pour tout X, Y, si $X \to Y$ alors $Y \to X$). [Aide : Le graphe comporte huit configurations.]*

SOLUTION. **[4 pts** $|\sum 42.0]$ Le graphe $G_{(3,2,1)}$ n'est pas symétrique car $(3, 1, 0) \to (4, 0, 0)$ mais $(4, 0, 0) \not\to (3, 1, 0)$.



Pour trouver un chemin d'une configuration S à T dans G_B avec le moins de transvasements possibles, on se propose d'utiliser l'algorithme A* vu en cours. Pour celà, on a besoin de générer tous les voisins d'un sommet donné.

Question 14. *Écrivez la fonction* `int successors(config X, config L[])` *qui étant donnée une configuration* `X` *de* G_B *renvoie dans* `L` *la liste des successeurs de* `X`*, c'est-à-dire toutes les configurations* `Y` *de* G_B *telles que* `X`→`Y`*. En valeur de retour, la fonction doit renvoyer le nombre de successeurs de* `X`*. [Aide : Attention ! S'il y a un transvasement, on doit obtenir une configuration différente de* `X`*.]*

SOLUTION. **[3 pts** $|\sum 45.0$**]**

```
int successors(config X, config L[]){
  int s=0; // nombre de successeurs de X
  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      // transvasement i->j
      if(i==j) continue; // il faut des seaux différents
      if(X[i]==0) continue; // rien à prendre dans le seau i
      if(Y[i]==B[j]) continue; // rien à mettre dans le seau j
      s++; // un successeur de plus: L[s]
      for(int k=0; k<n; k++) L[s][k]=X[k]; // copie X vers L[s]
      if(X[i]+X[j]<=B[j]) L[s][i]=0, L[s][j] += X[i]; // vide le seau i
      else L[s][j]=B[j], L[s][i] -= B[j]-X[j]; // remplit le seau j
    }
  }
  return s;  // nombre de successeurs écris dans L
}
```

Question 15. *Quelle est la complexité de votre fonction ? Justifiez.*

SOLUTION. **[2 pts** $|\sum 47.0|$ C'est $O(n^3)$ à cause des trois boucles imbriquées (`for(i=0..n-1)`, `for(j=0..n-1)`, `for(k=0..n-1)`).

En combinaison avec A*, on propose d'utiliser l'heuristique h suivante : $h(X,T)$ est le nombre de seaux de la configuration X incorrectement remplis dans T. Par exemple, pour la batterie $B = (4,3,1)$, $X = (4,0,0)$ et $T = (2,2,0)$, nous avons $h(X,T) = 2$.

Question 16. *Démontrez que l'heuristique h n'est pas monotone.*

SOLUTION. **[2 pts** $|\sum 49.0|$ En reprenant l'exemple, $T = (2,2,0)$ et avec $X = (1,2,1)$, on a $X \to T$. Or $h(X,T) = 2 \not\leqslant 1 + h(T,T) = 1 + 0$. Donc h n'est pas monotone.

Question 17. *Proposez une variante de l'heuristique précédente afin que A* trouve un plus court chemin de S à T, s'il existe.*

SOLUTION. **[2 pts** $|\sum 51.0|$ Il suffit de considérer l'heuristique $h'(X,T) = h(X,T)/2$. Il vient alors qu'avec un seul arc $X \to Y$, on ne peut pas diminuer plus d'une unité l'heuristique, et donc $h'(Y,T) \geqslant h'(X,T) - 1$, ce qui revient à dire que $h'(X,T) \leqslant 1 + h'(Y,T)$ pour tout $X \to Y$, ce qui est la définition de monotonie. Bien sûr, l'heuristique $h'(X,T) = 0$, bien que monotone, n'est pas considérée comme une variante de h.

FIN.