VIGEANT, Dominique. Matricule 20129080.

Question 1. (10 points)

a) Il faut mettre la sous-liste la plus longue puisque ça nous garantit que la pile n'aura jamais plus que $\lg N$ éléments.

b) Nous savons que dans un arbre Full, $L = N + 1$, où L est le nombre de feuilles et N est le nombre de noeuds internes. Pour rendre cet arbre non-Full, on peut soit enlever une feuille, soit en ajouter une. Si on enlève une feuille, le nombre de noeuds internes reste le même (puisque si ça en fait un de moins, ça voudrait dire que l'arbre n'était pas Full). On se retrouve avec $L = N$ donc c'est impossible d'avoir $L = N + 1$ dans ce cas. Si on ajoute une feuille, le nombre de noeuds internes augmente de 1 et le nombre de feuilles reste le même puisqu'on change une feuille en noeud mais on ajoute une feuille, ce qui donne $L = N$ encore une fois. Ainsi, si un arbre binaire non-vide n'est pas Full il est impossible que le nombre de feuilles soit égal au nombre de noeuds internes plus un.

c)

$$\begin{aligned} & O(k + (N - k)\log k) \\ = \; & O(k + N\log k - k\log k) \end{aligned}$$

k et $k\log k$ sont négligeables, on peut donc les enlever.

$$= O(N\log k)$$

Ici, $\log k$ est un facteur de N mais n'est pas une constante alors on le garde.

d) Dans le cas du sac les éléments n'ont pas besoin d'être triés. Nous avons donc quelques données avec un certains nombres d'opérations à effectuer, dont le tri ne fait pas partie. Ça c'est notre problème. Ici, la méthode jugée la meilleure était le tableau contigu. Dans le problème TAD *Liste*, on a des données avec quelques opérations, dont le tri peut faire partie. La méthode à choisir est moins claire puisque le tableau contigu et la liste chaînée ont chacun leurs avantages.

Question 2. (10 points)

a) Si la clé à chercher est au niveau λ, comme on commence au niveau 0, on cherche les niveau 0 à $\lambda - 1$ inclusivement sans trouver la clé. Cela fait donc 2 comparaisons à chaque niveau pour λ niveaux. Rendu au niveau λ on se demande "la clef de recherche est-elle égale à la clef dans le noeud ?" et la réponse est positive, nous avons donc fait une comparaison supplémentaire. Ainsi, on a fait un total de $2\lambda + 1$ comparaisons.

Si on a un arbre particulier à N noeuds mais qu'on ne sait pas la profondeur de la clé, on utilise la moyenne de profondeur des N noeuds de l'arbre. Puisque $I(N)$ est la somme des profondeurs des noeuds de l'arbre, $\frac{I(N)}{N}$ est la moyenne des profondeurs des noeuds de l'arbre. Il faut donc faire en moyenne $C_N = 2\left(\frac{I(N)}{N}\right)+1 = \frac{2I(N)+N}{N}$ comparaisons pour une recherche réussite.

b) i. Pour un arbre parfaitement balancé τ_{gentil} à d niveaux et $N = 2^{d+1}-1$ noeuds, la somme de profondeur de tous les noeuds d'un niveau i tel que $0 \leq i \leq d$ est $i2^i$. Pour avoir la longueur de chemin interne, il ne reste qu'à additionner la valeur correspondante pour chaque niveau. On a que

$$I(N) = \sum_{i=0}^{d} i2^i = 2(2^d d - 2^d + 1) = 2 \cdot d2^d - 2 \cdot 2^d + 2 = d2^{d+1} - 2^{d+1} + 2 = 2^{d+1}(d-1) + 2 = (N+1)(\lg(N+1) - 2) + 2$$

Ainsi, l'approximation donnée manque 2 de profondeur, mais pour un N très grand c'est valide comme approximation.

b) ii. Selon la valeur approximative donnée, le terme dominant serait $N\lg N$. On a donc que

$$I(N) = N\lg N \Rightarrow C_N = \frac{2N\lg N + N}{N} \Rightarrow \lim_{N\to\infty} C_N = \lim_{N\to\infty} \frac{2N\lg N + N}{N} =$$

$$\lim_{N\to\infty} \frac{2 \cdot \frac{d}{dN}(N\lg N) + 1}{1} \quad Hopital = \lim_{N\to\infty} \frac{2(\lg N + \frac{1}{\ln 2}) + 1}{1} = \infty$$

c) Puisque l'arbre $\tau_{mechant}$ n'est qu'une liste linéaire de longueur N, où le i^e noeud aura $i - 1$ comme profondeur,

$$I(N) = \sum_{i=0}^{N-1} i = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2}$$

$$C_N = \frac{2 \cdot I(N) + N}{N} = \frac{2 \cdot \frac{(N-1)N}{2} + N}{N} = \frac{(N-1)N + N}{N} = \frac{N(N-1+1)}{N} = N$$

d)

$$\begin{aligned} C_N &= \frac{2 \cdot I(N) + N}{N} \\ &= \frac{2 \cdot \frac{D(N)}{N}}{N} + 1 \\ &= \frac{2 \cdot \frac{D(N)}{N+1}}{N} + 1 \\ &= 4\frac{1}{N}\sum_{i=3}^{N+1} \frac{1}{i} + 1 \\ &= \frac{4H_{N+1} + N}{N} \\ &= \frac{4\ln(N+1) + N}{N} \end{aligned}$$

On voit que $\tau_{mechant}$ prend beaucoup plus de temps que τ_{moyen}, qui lui prend moins de temps que τ_{gentil}.

Question 3. (10 points)

a) Selon les propriétés du monceau, la clé la plus grande sera l'une des feuilles. Pour trouver l'enfant gauche d'un noeud i, on fait $2i$, pour trouver l'enfant droit on fait $2i+1$. Donc pour trouver le parent d'une noeud $i \geq 2$, on peut faire $\lfloor \frac{i}{2} \rfloor$. Pour trouver le nombre de feuilles, on peut donc trouver le parent de la dernière feuille (du dernier noeud). Cela donne le nombre de noeuds internes puisque le parent de la dernière feuille sera le dernier noeud interne. Donc pour un arbre de N noeuds, le Ne noeud sera la dernière feuille et donc $N - \lfloor \frac{N}{2} \rfloor = \lceil \frac{N}{2} \rceil = F$, où F est le nombre de feuilles de l'arbre. Pour trouver le clé la plus grande on peut comparer toutes les feuilles en partant de l'élément $\lfloor \frac{N}{2} \rfloor + 1$. Puisqu'on fait $\lceil \frac{N}{2} \rceil$ comparaisons, ça coûte $O(N)$ à trouver. Le pire scénario serait le cas d'un arbre binaire parfait (un arbre où pour une hauteur h, il y a $2^{h+1} - 1$ noeuds) et que l'élément avec la priorité la plus basse se trouve se trouve dans le dernier noeud (le dernier élément du tableau).

b) i.

$$\begin{aligned}
C'_N &= \frac{2\ E(N)}{N+1} \\
&= \frac{2\ (I(N)+2N)}{N+1} \\
&= \frac{2I(N)+4N)}{N+1} \\
&= \left(\frac{N}{N+1}\right)\left(\frac{2I(N)+N+3N}{N}\right) \\
&= \left(\frac{N}{N+1}\right)\left(\frac{2I(N)+N}{N}+3\right) \\
&= \left(\frac{N}{N+1}\right)(C_N+3)
\end{aligned}$$

b) ii.

$$\begin{aligned}
C'_N &= \left(\frac{N}{N+1}\right)(C_N+3) \\
\Rightarrow C_N &= \frac{C'_N(N+1)}{N} - 3 \\
&= \frac{(4H_{N+1}-4)(N+1)}{N} - 3 \\
&= \frac{(4(H_N+\frac{1}{N+1})-4)(N+1)}{N} - 3 \\
&= \frac{4(H_N+\frac{1}{N+1}-1)(N+1)}{N} - 3 \\
&= \frac{4(NH_N+\frac{N}{N+1}-N+H_N+\frac{1}{N+1}-1)}{N} - 3 \\
&= \frac{4(H_N(N+1)+\frac{N+1}{N+1}-(N+1))}{N} - 3 \\
&= \frac{4(H_N(N+1)-(N+1)+1)}{N} - 3 \\
&= 4\left(\frac{H_N(N+1)-N}{N}\right) - 3 \\
&= 4\left(\frac{H_N(N+1)}{N}-1\right) - 3 \\
&= 4\left(\frac{NH_N+H_N}{N}\right) - 7 \\
&= 4\left(1+\frac{1}{N}\right)H_N - 7
\end{aligned}$$

C'est similaire.