Structures de données
TD - 2

1 Recherche d'une valeur dans une tableau non-trié

Algorithm 1 Recherche dans un tableau non-trié

```
1: procedure FindValue(T[ ], v)
2:     n ← getNbElement(T)
3:     for i ← 1, n do
4:         if T[i] == v then
5:             return i
6:         end if
7:     end for
8:     return −1
9: end procedure
```

1.1. Dans l'algorithme 1, quelles est la condition pour que le meilleur cas se produise ?

1.2. Pour le meilleur cas de cette algorithme,

(a) Quel est son coût en temps ?

(b) Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

1.3. Dans l'algorithme 1, quelle est la condition pour que le pire cas se produise ?

1.4. Pour le pire cas de cette algorithme,

(a) Quel est son coût en temps ?

(b) Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

1.5. Dans l'algorithme 1, quelle hypothèse peut-on faire pour le calcul du cas moyen ?

1.6. Pour le cas moyen de cette algorithme,

(a) Quel est son coût en temps ?

(b) Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

2 Recherche d'une valeur dans une tableau trié

Algorithm 2 Recherche par dichotomie

```
1: procedure FindValue(T[ ], v)
2:     start ← 1
3:     end ← getNbElement(T)
4:     while start ≤ end do
5:         i ← ⌊(start+end)/2⌋
6:         if T[i] == v then
7:             return i
8:         else if T[i] < v then
9:             start ← i + 1
10:        else
11:            end ← i − 1
12:        end if
13:    end while
14:    return −1
15: end procedure
```

2.1. Dans l'algorithme 2, quelles est la condition pour que le meilleur cas se produise ?

2.2. Pour le meilleur cas de cette algorithme,

(a) Quel est son coût en temps ?

(b) Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

2.3. Dans l'algorithme 2, quelle est la condition pour que le pire cas se produise ?

2.4. Pour le pire cas de cette algorithme,

(a) Quel est son coût en temps ?

(b) Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

3 Recherche des k plus petits éléments.

```c
#include <stdio.h>
#include <stdlib.h>


size_t * rkppe(float * tab, size_t n, size_t k){
        size_t * idx = malloc(k*sizeof(size_t));
        idx[0] = 0;
        for(size_t i = 1 ; i < n ; i++){
                size_t k_prim = i<k?i:k;
                size_t j = 0;
                while(j < k_prim && tab[i] >= tab[idx[j]]) j++;
                for(size_t p = k-1 ; p > j ; p--){
                        idx[p] = idx[p-1];
                }
                if(j<k)
                        idx[j] = i;
        }
        return idx;
}

int main(int argc, char*argv[]){
        float tab[] = {9.0, 5.0, 2.0, 12.0, 11.0, 4.0, 3.0, 25.0, 3.0, 8.0};
        size_t * idx = rkppe(tab, 10, 4);

        for(size_t i = 0; i< 4; i++)
                printf("%lu␣%f,␣", idx[i], tab[idx[i]]);
        printf("\n");
        free(idx);
}
```