Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

HE IG VD

Recueil d'exercices d'Algorithmes et Structures de Données (ASD)

Version 0.2

28 mars 2023

Table des matières

Chapitre 1 : Introduction

Exercice 1.1 Complexités (1)

Quelle est la complexité des extraits de code suivants en fonction du paramètre N ?

1.	`int k = 0;` `for(int i = 0; i < N; ++i)` `  ++k;`	
2.	`int k = 0;` `for(int i = 0; i*i < N; ++i)` `  ++k;`	
3.	`int k = 0;` `for(int i = 1; i < N; i *= 2)` `  ++k;`	
4.	`int k = 0;` `for(int i = 0; i < N; ++i)` `  for(int j = 0; j < N; ++j)` `  ++k;`	
5.	`int k = 0;` `for(int i = 0; i < N; ++i)` `  for(int j = 1; j < N; j *= 2)` `  ++k;`	
6.	`int k = 0;` `for(int i = 0; i < N; ++i)` `  for(int j = i; j < N; ++j)` `  ++k;`	

Exercice 1.2 Complexités (2)

Quelle est la complexité des extraits de code suivants en fonction des deux paramètres M et N ?

1.	`int k = 0;` `for(int i = 0; i < N or i < M; ++i)` `  ++k;`	
2.	`int k = 0;` `for(int i = 0; i< N and i < M; ++i)` `  ++k;`	
3.	`int k = 0;` `for (int i = 1; i < N; i *= 2)` `  for (int j = 1; j < M; ++j)` `  ++k;`	
4.	`int k = 0;` `for(int i = 0; i < N; ++i) {` `  ++k;` `  for (; i < M; ++i)` `  ++k;` `}`	
5.	`int k = 0;` `for(int i = N; i < M; ++i)` `  ++k;`	
6.	`int k = 0;` `for(int i = 0; i < N; ++i) {` `  ++k;` `  for (int j = i; j < M; ++j)` `  ++k;` `}`	

Exercice 1.3 Complexités : pire cas / meilleur cas / en moyenne

Quelle est la complexité des extraits de code suivants dans le pire cas, meilleur cas et en moyenne ?

1.	`int k = 0;` `for(int i = 0; i < N; ++i) {` `  if(rand() % N == 0) {` `  for(int j = 0; j < N; ++j)` `  ++k;` `  } else {` `  ++k;` `  }` `}`	
2.	`int k = 0;` `for(int i = 0; i < N; ++i) {` `  if(rand() % 2) {` `  for(int j = 0; j < N; ++j)` `  ++k;` `  } else {` `  ++k;` `  }` `}`	
3.	`int k = 0;` `for(int i = 0; i < N; ++i) {` `  int a = rand() % N;` `  if(a*a < N) {` `  for(int j = 0; j < N; ++j)` `  ++k;` `  } else {` `  ++k;` `  }` `}`	
4.	`int k = 0;` `for(int i = 1; i < N; i *= 2) {` `  int a = rand() % N;` `  for(int j = 0; j < a; ++j)` `  ++k;` `}`	

5.	``` int k = 0; for(int i = 1; i < N; ++i) { if(rand() % 2 == 0) { for (int j = 0; j < M; ++j) ++k; } else { for (int j = 0; j < N; ++j) ++k; } } ```	

Exercice 1.4 `std::lower_bound, std::upper_bound`

Qu'affichent les extraits de code suivants ? Si une valeur est indéterminée, remplacez-là par le symbole '?'.

1.
```
vector v = { 1, 2, 3, 5, 7, 11, 13, 17 };
cout << *lower_bound(v.begin(),v.end(),0) << " ";
cout << *lower_bound(v.begin(),v.end(),1) << " ";
cout << *lower_bound(v.begin(),v.end(),3) << " ";
cout << *lower_bound(v.begin(),v.end(),4) << " ";
cout << *lower_bound(v.begin(),v.end(),17) << " ";
cout << *lower_bound(v.begin(),v.end(),18) << endl;
```

2.
```
vector v = { 1, 2, 3, 5, 7, 11, 13, 17 };
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),0)) << " ";
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),1)) << " ";
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),3)) << " ";
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),4)) << " ";
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),17)) << " ";
cout << distance(v.begin(),lower_bound(v.begin(),v.end(),18)) <<
endl;
```

3.
```
vector v = { 1, 2, 3, 5, 7, 11, 13, 17 };
cout << *upper_bound(v.begin(),v.end(),0) << " ";
cout << *upper_bound(v.begin(),v.end(),1) << " ";
cout << *upper_bound(v.begin(),v.end(),3) << " ";
cout << *upper_bound(v.begin(),v.end(),4) << " ";
cout << *upper_bound(v.begin(),v.end(),17) << " ";
cout << *upper_bound(v.begin(),v.end(),18) << endl;
```

4.
```
vector v = { 1, 2, 3, 5, 7, 11, 13, 17 };
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),0)) << " ";
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),1)) << " ";
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),3)) << " ";
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),4)) << " ";
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),17)) << " ";
cout << distance(v.begin(),upper_bound(v.begin(),v.end(),18)) <<
endl;
```

Exercice 1.5 `std::equal_range`

Qu'affichent les extraits de code suivants ? Si une valeur est indéterminée, remplacez-là par le symbole '?'.

1.
```
vector v = {1, 2, 2, 3, 5, 5, 7, 7, 11, 13};
auto r = equal_range(v.begin(), v.end(), 5);
cout << distance(v.begin(), r.first) << " ";
cout << distance(r.first, r.second) << " ";
cout << *r.first << " " << *r.second << endl;
```

2.
```
vector v = {1, 2, 2, 3, 5, 5, 7, 7, 11, 13};
auto r = equal_range(v.begin(), v.end(), 8);
cout << distance(v.begin(), r.first) << " ";
cout << distance(r.first, r.second) << " ";
cout << *r.first << " " << *r.second << endl;
```

3.
```
vector v = {1, 2, 2, 3, 5, 5, 7, 7, 11, 13};
auto r = equal_range(v.begin(), v.end(), 13);
cout << distance(v.begin(), r.first) << " ";
cout << distance(r.first, r.second) << " ";
cout << *r.first << " " << *r.second << endl;
```

4.
```
vector v = {1, 2, 2, 3, 5, 5, 7, 7, 11, 13};
auto r = equal_range(v.begin(), v.end(), 17);
cout << distance(v.begin(), r.first) << " ";
cout << distance(r.first, r.second) << " ";
cout << *r.first << " " << *r.second << endl;
```

Solutions

Exercice 1.1 – Complexités (1)

1. $O(N)$
2. $O(\sqrt{N})$
3. $O(\log N)$
4. $O(N^2)$
5. $O(N \log N)$
6. $O(N^2)$

Exercice 1.2 – Complexités (2)

1. $O(\max(M, N))$, mais $O(M + N)$ est aussi acceptable
2. $O(\min(M, N))$
3. $O(\mathrm{M} \log N)$
4. $O(\max(M, N))$, mais $O(M + N)$ est aussi acceptable
5. $O(\max(0, M - N))$
6. $O(M^2 + N)$ si $M < N$, $O(2MN - N\text{^}2)$ sinon

Exercice 1.3 – Complexités : pire cas / meilleur cas / en moyenne

1. Pire : $O(N^2)$ – Meilleur : $O(N)$ – Moyenne : $O(N)$
2. Pire : $O(N^2)$ – Meilleur : $O(N)$ – Moyenne : $O(N^2)$
3. Pire : $O(N^2)$ – Meilleur : $O(N)$ – Moyenne : $O(N^{1.5})$
4. Pire : $O(N \log N)$ – Meilleur : $O(\log N)$ – Moyenne : $O(N \log N)$
5. Pire : $O(N \max(M, N)$ – Meilleur : $O(N \min(M, N))$ – Moyenne : $O(N(N + M)$

Exercice 1.4 – lower_bound et upper_bound

1. `1 1 3 5 17 ?`
2. `0 0 2 3 7 8`
3. `1 2 5 5 ? ?`
4. `0 1 3 3 8 8`

Exercice 1.5 – equal_range

1. 4 2 5 7
2. 8 0 11 11
3. 9 1 13 ?
4. 10 0 ? ?

Chapitre 2 : Récursivité

Exercice 2.1 Tracer un appel récursif

Qu'affichent les fonctions suivantes si on les appelle avec la valeur 4 en paramètre ?

1.	`void f1(int n) {` `  if(n > 0) {` `  cout << n << " ";` `  f1(n-1);` `  }` `}`	
2.	`void f2(int n) {` `  if(n > 0) {` `  f2(n-1);` `  cout << n << " ";` `  }` `}`	
3.	`void f3(int n) {` `  cout << n << " ";` `  if(n > 0) {` `  f3(n-1);` `  }` `}`	
4.	`void f4(int n) {` `  if(n > 0) {` `  f4(n-1);` `  }` `  cout << n << " ";` `}`	

5.	`void f5(int n) {` `  cout << n << " ";` `  if(n < 100) {` `  f5(n*2);` `  }` `}`	
6.	`void f6(int n) {` `  if(n < 100) {` `  f6(n*n/2);` `  cout << n << " ";` `  }` `}`	
7.	`void f7(int n) {` `  if(n > 1) {` `  cout << n << " ";` `  f7(n-1);` `  }` `  cout << n << " ";` `}`	
8.	`void f8(int n) {` `  cout << n << " ";` `  if(n > 0) {` `  f8(n-2);` `  }` `  cout << n << " ";` `}`	

Exercice 2.2 Tracer deux appels récursifs

Qu'affichent les fonctions suivantes si on les appelle avec la valeur 3 en paramètre ?

1.	`void f1(int n) {` `  if(n > 0) {` `  cout << n << " ";` `  f1(n-1);` `  f1(n-1);` `  }` `}`	
2.	`void f2(int n) {` `  if(n > 0) {` `  f2(n-1);` `  cout << n << " ";` `  f2(n-1);` `  }` `}`	
3.	`void f3(int n) {` `  if(n > 0) {` `  f3(n-1);` `  f3(n-1);` `  cout << n << " ";` `  }` `}`	
4.	`void f4(int n) {` `  if(n > 0) {` `  cout << n << " ";` `  f4(n-1);` `  f4(n-2);` `  }` `}`	

5.	`void f5(int n) {` `  if(n > 0) {` `  f5(n-2);` `  f5(n-1);` `  }` `  if(n >= 0)` `  cout << n << " ";` `}`	
6.	`void f6(int n) {` `  if(n)` `  f6(n-1);` `  cout << n << " ";` `  if(n > 1)` `  f6(n-2);` `}`	

Exercice 2.3 Tracer plusieurs appels récursifs

Qu'affichent les fonctions suivantes si on les appelle avec la valeur 3 en paramètre ?

1.	`void f1(int n) {` `  cout << n << " ";` `  for(int i = 0; i < n; ++i)` `  f1(i);` `}`	
2.	`void f2(int n) {` `  for(int i = 0; i < n; ++i)` `  f2(i);` `  cout << n << " ";` `}`	
3.	`void f3(int n) {` `  for(int i = 0; i < n; ++i) {` `  cout << n << " ";` `  f3(i);` `  }` `}`	
4.	`void f4(int n) {` `  cout << n << " ";` `  while(n) f4(--n);` `}`	
5.	`void f5(int n) {` `  while(n) f5(--n);` `  cout << n << " ";` `}`	

6.	`int f6(int n) {` `  int a = 1;` `  if(n) {` `  a = f6(n - 1);` `  a += f6(n - 1);` `  cout << a << " ";` `  }` `  return a;` `}`	
7.	`int f7(int n) {` `  int a = 1;` `  if(n) {` `  a = f7(n / 2);` `  a += f7(n / 2);` `  }` `  cout << a << " ";` `  return a;` `}`	

Exercice 2.4 Complexité code récursif

Quelle est la complexité des fonctions ci-dessous en fonction du paramètre n ?

1.	`int f1(int n) {` `  if(n > 0)` `  return f1(n-1) + f1(n-1);` `  else` `  return 1;` `}`	
2.	`int f2(int n) {` `  if(n > 0)` `  return 2 * f2(n-1);` `  else` `  return 1;` `}`	
3.	`int f3(int n) {` `  if(n > 0)` `  return f3(n/2) + f3(n/2);` `  else` `  return 1;` `}`	
4.	`int f4(int n) {` `  if(n > 0)` `  return 2*f4(n/2);` `  else` `  return 1;` `}`	
5.	`int f5(int n) {` `  if(n > 1)` `  return f5(n-1) + f5(n-2);` `  else` `  return n;` `}`	
6.	`int f6(int n) {` `  if(n > 0) {` `  int a = 0;` `  for(int i = 0; i < n; ++i)` `  a += f6(n-1);` `  return a;` `  }` `  else` `  return 1;` `}`	

Exercice 2.5 Triangle de lettres

En n'utilisant pas d'instruction de boucle - mais uniquement la récursivité – écrivez la fonction `void triangle(int n);` telle que l'appel à `triangle(5);` affiche les 6 lignes suivantes

```
A
BAB
CBABC
DCBABCD
EDCBABCDE
FEDCBABCDEF
```

Conseil : écrivez une fonction auxiliaire récursive qui affiche une ligne.

Exercice 2.6 Puissance de b

Ecrivez une fonction `double puissance(double b, unsigned n);` qui calcule b^n

1. En utilisant la relation récurrente $b^n = b \cdot b^{n-1}$
2. En utilisant la relation récurrent $b^n = b^{\frac{n}{2}} \cdot b^{\frac{n}{2}}$ quand b est pair et $b^n = b \cdot b^{\frac{n}{2}} \cdot b^{\frac{n}{2}}$ quand b est impair

Quelle est la complexité de chacune de ces deux fonctions ?

Solutions

Exercice 2.1 – Tracer un appel récursif

1. 4 3 2 1
2. 1 2 3 4
3. 4 3 2 1 0
4. 0 1 2 3 4
5. 4 8 16 32 64 128
6. 32 8 4
7. 4 3 2 1 2 3 4
8. 4 2 0 0 2 4

Exercice 2.2 – Tracer deux appels récursifs

1. 3 2 1 1 2 1 1
2. 1 2 1 3 1 2 1
3. 1 1 2 1 1 2 3
4. 3 2 1 1
5. 0 1 0 0 1 2 3
6. 0 1 2 0 3 0 1

Exercice 2.3 – Tracer plusieurs appels récursifs

1. 3 0 1 0 2 0 1 0
2. 0 0 1 0 0 1 2 3
3. 3 3 1 3 2 2 1
4. 3 2 1 0 0 1 0 0
5. 0 0 0 0 0 0 0 0
6. 2 2 4 2 2 4 8
7. 1 1 2 1 1 2 4

Exercice 2.4 - Complexité code récursif

1. $O(2^n)$
2. $O(n)$
3. $O(n)$
4. $O(\log n)$
5. $O(\phi^n)$ avec $\phi = (\sqrt{5}+1)/2$
6. $O(n!)$

Exercice 2.5 – Triangle de lettres

```
#include <iostream>

using namespace std;

void ligne(int n) {
   if(n) {
      cout << char('A' + n);
      ligne(n-1);
   }
   cout << char('A'+n);
}

void triangle(int n) {
   if(n)
      triangle(n-1);
   ligne(n);
   cout << endl;
}

int main () {
   triangle(5);
}
```

Exercice 2.6 – Puissance de b

1. En utilisant $b^n = b \cdot b^{n-1}$ et $b^0 = 1$,

   ```
   double puissance1(double b, unsigned n) {
      if(n)
         return b* puissance1(b,n-1);
      else
         return 1.;
   }
   ```

   Cette fonction a une complexité $O(n)$

2. En utilisant $b^n = b^{\frac{n}{2}} \cdot b^{\frac{n}{2}}$ quand b est pair, $b^n = b \cdot b^{\frac{n}{2}} \cdot b^{\frac{n}{2}}$ quand b est impair et $b^0 = 1$,

   ```
   double puissance2(double b, unsigned n) {
      if(n == 0) return 1.;
      double p2 = puissance2(b,n/2);
      return p2 * p2 * (n % 2 ? b : 1.);
   }
   ```

Cette fonction a une complexité $O(\log n)$.

Attention à ne pas écrire cette fonction avec deux appels récursifs. En effet, la fonction suivante a une complexité $O(n)$

```
double puissance3(double b, unsigned n) {
   if(n == 0) return 1.;
   return puissance3(b,n/2) * puissance3(b,n/2) * (n % 2 ? b : 1.);
}
```

Chapitre 3 : Tris

Exercice 3.1 Tri à bulle

Effectuez le tri à bulle des tableaux suivants. Affichez l'état du tableau après chaque échange.

3	5	2	4	1	6

4	5	2	1	6	3

Exercice 3.2 Tri par sélection

Effectuez le tri par sélection des tableaux suivants. Affichez l'état du tableau après chaque échange, y compris si l'échange se fait avec lui-même.

3	5	2	4	1	6

4	5	2	1	6	3

Exercice 3.3 Tri par insertion

Effectuez le tri par insertion des tableaux suivants. Affichez l'état du tableau à chaque fois que le contenu du tableau change. Par exemple, pour le tableau [3, 1, 2], on aurait

3	1	2
3	3	2
1	3	2
1	3	3
1	2	3

2	1	5	3	4

4	5	2	1	3

Exercice 3.4 Tri fusion

Effectuez le tri fusion des tableaux suivants, en supposant que celui-ci est mis en oeuvre avec deux appels récursifs. Indiquez l'état du tableau après chaque étape de fusion

6	2	5	1	8	3	7	4

8	2	7	3	6	5	1	1

Exercice 3.5 Partition rapide

Effectuez la partition du tri rapide sur les tableau suivants en utilisant l'élément souligné comme pivot. Indiquez l'état du tableau après chaque échange.

6	8	5	1	2	3	7	4

5	2	3	2	4	6	2	1

5	4	3	3	2	3	1	6

Exercice 3.6 Tri rapide

Effectuez le tri rapide sur les tableau suivants en choisissant l'élément le plus à gauche comme pivot lors des partitions, et en partitionnant d'abord le plus petit des sous-tableaux, (celui de gauche en cas tailles égales). Affichez l'état du tableau après chaque partition d'un tableau de plus de 1 élément.

6	7	5	1	2	3	8	4

3	1	4	2	5	6	1	2

5	1	3	3	2	3	4	6

Exercice 3.7 Sélection rapide

Effectuez la sélection rapide de la médiane sur les tableau suivants en choisissant l'élément le plus à gauche comme pivot lors des partitions. Affichez l'état du tableau après chaque partition d'un tableau de plus de 1 élément.

6	7	5	1	2	3	4

3	7	1	5	6	2	4

3	5	1	6	2	4	7

Exercice 3.8 Devinez le tri utilisé

On effectue le tri d'un tableau de nombres avec 5 algorithmes (bulle, sélection, insertion, fusion, rapide) et l'on observe un état intermédiaire du tableau. Reliez chaque état intermédiaire au tri correspondant.

3.8.1 Tableau à trier : `4 2 6 3 5 9 1 7 8 0`

`0 1 2 3 5 9 6 7 8 4`	○	○ Tri à bulles
`2 3 4 5 6 1 9 0 7 8`	○	○ Tri par sélection
`2 3 4 5 6 9 1 7 8 0`	○	○ Tri par insertion
`0 2 1 3 4 9 6 7 8 5`	○	○ Tri fusion
`2 3 1 4 5 0 6 7 8 9`	○	○ Tri rapide

3.8.2 Tableau à trier : `9 8 7 6 5 4 3 2 1 0`

`4 5 6 7 8 9 3 2 1 0`	○	○ Tri à bulles
`6 5 4 3 2 1 0 7 8 9`	○	○ Tri par sélection
`1 0 7 6 5 4 3 2 8 9`	○	○ Tri par insertion
`8 9 5 6 7 4 3 2 1 0`	○	○ Tri fusion
`0 1 2 3 5 4 6 7 8 9`	○	○ Tri rapide

3.8.3 Tableau à trier : `3 5 7 9 1 2 4 6 8 0`

`0 2 1 3 7 5 4 6 8 9`	○	○ Tri à bulles
`1 2 3 5 7 9 4 6 8 0`	○	○ Tri par sélection
`3 1 2 4 5 6 7 0 8 9`	○	○ Tri par insertion
`0 1 2 3 4 7 5 6 8 9`	○	○ Tri fusion
`1 3 5 7 9 0 2 4 6 8`	○	○ Tri rapide

Exercice 3.9 stable_sort

Etant donné les fonctions de comparaison suivantes

```
bool tri_par_unite(int a, int b) {
   return a % 10 < b % 10;
}

bool tri_par_dizaine (int a, int b) {
   return a / 10 < b / 10;
}
```

Que contient le vecteur v après l'exécution des codes suivant ? Indiquez « indeterminé » si le standard C++ ne garanti pas ce contenu

1.	`vector v{12, 43, 37, 56, 34, 57, 51, 33};` `std::stable_sort(v.begin(), v.end(),` `tri_par_unite);` `std::stable_sort(v.begin(), v.end(),` `tri_par_dizaine);`	
2.	`vector v{12, 43, 37, 56, 34, 57, 51, 33};` `std::stable_sort(v.begin(), v.end(),` `tri_par_dizaine);` `std::stable_sort(v.begin(), v.end(),` `tri_par_unite);`	
3.	`vector v{12, 43, 37, 56, 34, 57, 51, 33};` `std::sort(v.begin(), v.end(),` `tri_par_unite);` `std::stable_sort(v.begin(), v.end(),` `tri_par_dizaine);`	
4.	`vector v{12, 43, 37, 56, 34, 57, 51, 33};` `std::stable_sort(v.begin(), v.end(),` `tri_par_unite);` `std::sort(v.begin(), v.end(),` `tri_par_dizaine);`	
5.	`vector v{12, 43, 37, 56, 34, 57, 51, 33};` `std::stable_sort(v.begin(), v.end(),` `tri_par_dizaine);` `std::sort(v.begin(), v.end(),` `tri_par_unite);`	

Exercice 3.10 qsort

Réécrivez le code ci-dessous en C en utilisant la fonction `qsort` de la librairie `"stdlib.h"` plutôt que la fonction C++ `std::sort` de la libraire `<algorithm>`.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

bool tri_par_unite(int a, int b) {
   return a % 10 < b % 10;
}

int main() {

  vector v{12, 43, 37, 56, 34, 57, 51, 33};
  std::sort(v.begin(), v.end(), tri_par_unite);
  for(int i : v)
    cout << i << " ";
  cout << endl;
}
```

Exercice 3.11 qsort (2)

Complétez ce programme en C pour qu'il affiche la liste des conseillers fédéraux par ordre alphabétique des partis, et pour les membres d'un même parti, par ordre alphabétique des noms de famille.

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"

typedef struct {
   char nom[32], prenom[32], parti[4];
} Personne;

int main(void) {
   Personne conseillers_federaux[] = {
          {.nom = "Berset", .prenom = "Alain", .parti = "PS"},
          {.nom = "Parmelin", .prenom = "Guy", .parti = "UDC"},
          {.nom = "Cassis", .prenom = "Ignazio", .parti = "PLR"},
          {.nom = "Amherd", .prenom = "Viola", .parti = "LC"},
          {.nom = "Keller-Sutter", .prenom = "Karin", .parti = "PLR"},
          {.nom = "Rosti", .prenom = "Albert", .parti = "UDC"},
          {.nom = "Baume-Schneider", .prenom = "Elisabeth", .parti = "PS"}
          };
   const int N = sizeof(conseillers_federaux) / sizeof (Personne);

   for(int i = 0; i < N; ++i)
      printf("%s %s (%s) \n",
             conseillers_federaux[i].nom,
             conseillers_federaux[i].prenom,
             conseillers_federaux[i].parti);
}
```

Exercice 3.12 Complexités tris de la STL

Quelle est la complexité des fonctions ci-dessous en fonction de N (et M)? Précisez cette complexité dans le pire cas et le cas moyen si elle varie en fonction du contenu du vecteur.

1.	`bool f1(vector<int> const& v, int val) {` `  size_t N = v.size();` `  if(is_sorted(v.begin(),v.end()))` `  return distance(v.begin(),` `  lower_bound(v.begin(),v.end(),val));` `  throw std::invalid_argument("v non trié");` `}`	

2.	`bool f2(vector<int> const& v, int val) {` `  size_t N = v.size();` `  assert(is_sorted(v.begin(),v.end()));` `  return distance(v.begin(),` `  lower_bound(v.begin(),v.end(),val));` `}`	
3.	`vector<int> f3(vector<int> const& v, vector<int>` `const&w) {` `  size_t N = v.size(), M = w.size();` `  vector<int> x(M+N);` `  merge(v.begin(),v.end(),` `  w.begin(),w.end(),` `  x.begin());` `  return x;` `}`	
4.	`array<int, 3> f4(vector<int>& v) {` `  sort(v.begin(),v.end());` `  return { v.front(), v[v.size()/2], v.back() };` `}`	
5.	`array<int, 3> f5(vector<int>& v) {` `  size_t N = v.size();` `  auto mid = v.begin() + N/2;` `  array<int,3> a;` `  nth_element(v.begin(),mid,v.end());` `  a[0] = *min_element(v.begin(), mid );` `  a[1] = *mid;` `  a[2] = *max_element(mid, v.end());` `  return a;` `}`	
6.	`array<int, 3> f6(vector<int>& v) {` `  stable_sort(v.begin(),v.end());` `  return { v.front(), v[v.size()/2], v.back() };` `}`	

Solutions

Exercice 3.1 – Tri à bulles

3 5 2 4 1 6 3 2 5 4 1 6 3 2 4 5 1 6 3 2 4 1 5 6 2 3 4 1 5 6 2 3 1 4 5 6 2 1 3 4 5 6 1 2 3 4 5 6	4 5 2 1 6 3 4 2 5 1 6 3 4 2 1 5 6 3 4 2 1 5 3 6 2 4 1 5 3 6 2 1 4 5 3 6 2 1 4 3 5 6 1 2 4 3 5 6 1 2 3 4 5 6

Exercice 3.2 – Tri par sélection

3 5 2 4 1 6 1 5 2 4 3 6 1 2 5 4 3 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6	4 5 2 1 6 3 1 5 2 4 6 3 1 2 5 4 6 3 1 2 3 4 6 5 1 2 3 4 6 5 1 2 3 4 5 6

Exercice 3.3 – Tri par insertion

2 1 5 3 4 2 2 5 3 4 1 2 5 3 4 1 2 5 5 4 1 2 3 5 4 1 2 3 5 5 1 2 3 4 5	4 5 2 1 3 4 5 5 1 3 4 4 5 1 3 2 4 5 1 3 2 4 5 5 3 2 4 4 5 3 2 2 4 5 3 1 2 4 5 3 1 2 4 5 5 1 2 4 4 5 1 2 3 4 5

Exercice 3.4 – Tri fusion

6 2 5 1 8 3 7 4 2 6 5 1 8 3 7 4 2 6 1 5 8 3 7 4 1 2 5 6 8 3 7 4 1 2 5 6 3 8 7 4 1 2 5 6 3 8 4 7 1 2 5 6 3 4 7 8 1 2 3 4 5 6 7 8	8 2 7 3 6 5 4 1 2 8 7 3 6 5 4 1 2 8 3 7 6 5 4 1 2 3 7 8 6 5 4 1 2 3 7 8 5 6 4 1 2 3 7 8 5 6 1 4 2 3 7 8 1 4 5 6 1 2 3 4 5 6 7 8

Exercice 3.5 – Partition rapide

6 8 5 1 2 3 7 4 6 8 4 1 2 3 7 5 3 8 4 1 2 6 7 5 3 2 4 1 8 6 7 5 3 2 4 1 5 6 7 8	5 2 3 2 4 6 2 1 5 2 3 1 4 6 2 2 2 2 3 1 4 6 5 2 2 1 3 2 4 6 5 2 2 1 2 2 4 6 5 3	5 4 3 3 2 3 1 6 5 4 3 6 2 3 1 3 1 4 3 6 2 3 5 3 1 3 3 6 2 4 5 3 1 3 2 6 3 4 5 3 1 3 2 3 3 4 5 6

Exercice 3.6 – Tri rapide

6 7 5 1 2 3 8 4 4 3 5 1 2 6 8 7 4 3 5 1 2 6 7 8 2 3 1 4 5 6 7 8 1 2 3 4 5 6 7 8	3 1 4 2 5 6 1 2 2 1 1 2 3 6 4 5 2 1 1 2 3 5 4 6 2 1 1 2 3 4 5 6 1 1 2 2 3 4 5 6 1 1 2 2 3 4 5 6	5 1 3 3 2 3 4 6 4 1 3 3 2 3 5 6 3 1 3 3 2 4 5 6 2 1 3 3 3 4 5 6 1 2 3 3 3 4 5 6

Exercice 3.7 – Sélection rapide

6 7 5 1 2 3 4 4 3 5 1 2 6 7 2 3 1 4 5 6 7	3 7 1 5 6 2 4 2 1 3 5 6 4 7 2 1 3 4 5 7 6	3 5 1 6 2 4 7 2 1 3 6 7 4 5 2 1 3 5 4 6 7 2 1 3 4 5 6 7

Exercice 3.8 - Devinez le tri utilisé

Tableau à trier : `4 2 6 3 5 9 1 7 8 0`

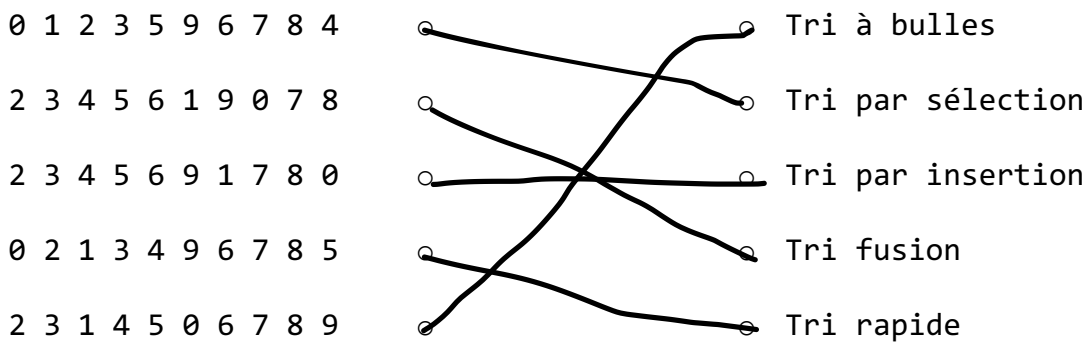


Tableau à trier : `9 8 7 6 5 4 3 2 1 0`

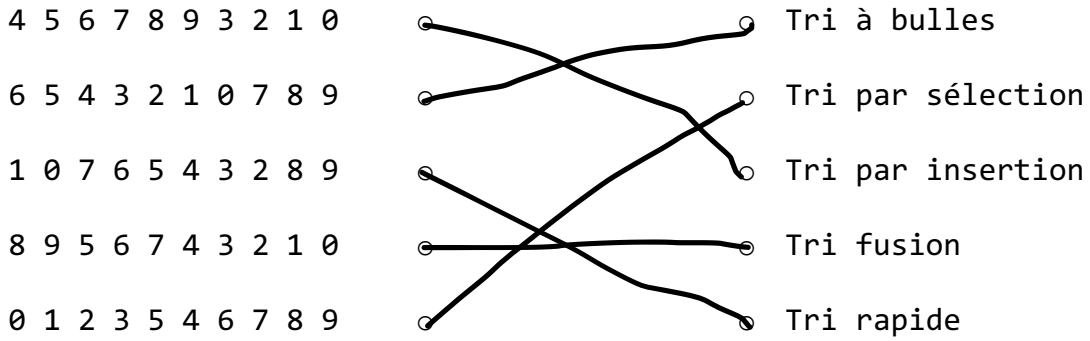
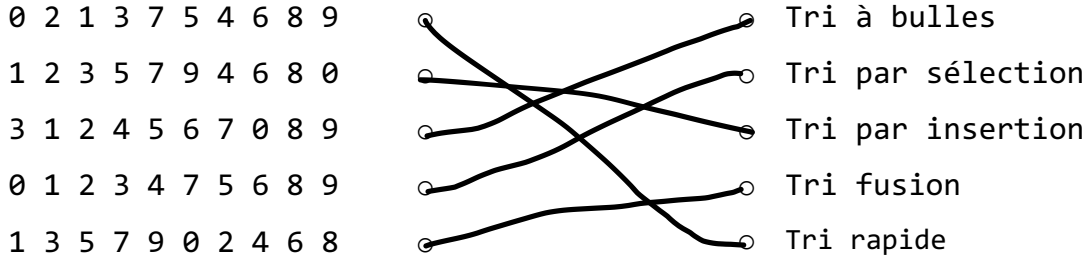


Tableau à trier : 3 5 7 9 1 2 4 6 8 0



Exercice 3.9 – stable_sort

1. 12 33 34 37 43 51 56 57
2. 51 12 33 43 34 56 37 57
3. 12 33 34 37 43 51 56 57
4. indéterminé
5. indéterminé

Exercice 3.10 – qsort

```c
#include "stdlib.h"
#include "stdio.h"

int tri_par_unite(void const* a, void const* b) {
   return *(int*)a % 10 - *(int*)b % 10;
}

int main(void) {
   int v[] = {12, 43, 37, 56, 34, 57, 51, 33};
   int n = sizeof(v) / sizeof(int);
   qsort(v, n, sizeof(int), tri_par_unite);
   for(int i = 0; i < n; ++i)
      printf("%d ",v[i]);
   printf("\n");
   return EXIT_SUCCESS;
}
```

Exercice 3.11 – qsort (2)

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"

typedef struct {
   char nom[32], prenom[32], parti[4];
} Personne;

int tri_par_parti_puis_nom(void const* a, void const* b) {
   int r = strcmp(((Personne*)a)->parti,((Personne*)b)->parti);
   return r ? r : strcmp(((Personne*)a)->nom,((Personne*)b)->nom);
}

int main(void) {
   Personne conseillers_federaux[] = {
          {.nom = "Berset", .prenom = "Alain", .parti = "PS"},
          {.nom = "Parmelin", .prenom = "Guy", .parti = "UDC"},
          {.nom = "Cassis", .prenom = "Ignazio", .parti = "PLR"},
          {.nom = "Amherd", .prenom = "Viola", .parti = "LC"},
          {.nom = "Keller-Sutter", .prenom = "Karin", .parti = "PLR"},
          {.nom = "Rosti", .prenom = "Albert", .parti = "UDC"},
          {.nom = "Baume-Schneider", .prenom = "Elisabeth", .parti = "PS"}
          };
   const int N = sizeof(conseillers_federaux) / sizeof (Personne);

   qsort(conseillers_federaux,N,sizeof(Personne),tri_par_nom);

   for(int i = 0; i < N; ++i)
      printf("%s %s (%s) \n", conseillers_federaux[i].nom,
        conseillers_federaux[i].prenom, conseillers_federaux[i].parti);
}
```

Exercice 3.12 - Complexités tris de la STL

1. $O(n)$, en raison de l'appel à `std::is_sorted`
2. $O(n)$ en mode debug, mais $O(\log n)$ en mode release
3. $O(n+m)$
4. $O(\mathrm{n}\log n)$ en moyenne, mais $O(n^2)$ dans le pire des cas
5. $O(n)$ en moyenne, mais $O(n^2)$ dans le pire des cas
6. $O(\mathrm{n}\log n)$ dans tous les cas

Chapitre X : Allocation dynamique

Exercice X.1 new et delete

Soit le code suivant

```
#include <iostream>

using namespace std;

class C {
   int i;
public:
   C() : i(0) { cout << "CD " << flush; }
   C(int i) : i(i) { cout << "C" << i << " " << flush; }
   ~C() { cout << "D" << i << " " << flush; }
};

int main() {
   auto p1 = f();
   auto p2 = f(2);
   g(p2);
   p2 = f(3);
   g(p1);
   g(p2);
}
```

Ecrivez les fonctions f, g et éventuelles surcharges de sorte que le programme affiche ce qui suit à l'exécution

```
CD C2 D2 C3 D0 D3
```

Exercice X.2 ::operator new, new(p), …

Soit le code suivant

```
#include <iostream>

using namespace std;

class C {
   int i;
public:
   C() : i(0) { cout << "CD " << flush; }
   C(int i) : i(i) { cout << "C" << i << " " << flush; }
   ~C() { cout << "D" << i << " " << flush; }
};

int main() {
  void *p1 = f(), *p2 = f();
  f(p1);
  f(p2,4);
  g((C*)p2);
  f(p2,1);
  g((C*)p1);
  ::operator delete(p1);
  f((C*)p2);
  ::operator delete(p2);
}
```

Ecrivez les fonctions f, g et éventuelles surcharges de sorte que le programme affiche ce qui suit à l'exécution

```
CD C4 D4 C1 D0 CD
```

Exercice X.3 vector::emplace et …

Soit la classe C suivante

```
class C {
   int i;
public:
   C() : i(0) { cout << "CD " << flush; }
   C(int i) : i(i) { cout << "C" << i << " " << flush; }
   C(C const& c) : i(c.i) { cout << "Cp" << i << " " << flush; }
   C& operator=(C const& c) { i = c.i; cout << "=" << i << " " << flush;
return *this; }
   ~C() { cout << "D" << i << " " << flush; }
};
```

Qu'affiche le code suivant à chacune de ses lignes

Code	Affichage
`int main() {`	
`  vector<C> v(2);`	
`  v.pop_back();`	
`  v.push_back(C(1));`	
`  v.clear();`	
`  v.emplace_back(2);`	
`  v.emplace_back(3);`	
`  v.front() = C(4);`	
`  v.pop_back();`	
`}`	

Solutions

Exercice X.1 new et delete

```
C* f() { return new C(); }

C* f(int i) { return new C(i); }

void g(C* p) { delete p; }
```

Exercice X.2 ::operator new, new(p), …

```
void* f() { return ::operator new(sizeof(C)); }

void f(void* p) { new(p) C; }

void f(void* p, int i) { new(p) C(i); }

void g(C* p) { p->~C(); }

// ou à partir de C++17, void g(C* p) { std::destroy_at(p); }
```

Exercice X.3 vector::emplace et …

`int main() {`	
`  vector<C> v(2);`	`CD CD`
`  v.pop_back();`	`D0`
`  v.push_back(C(1));`	`C1 Cp1 D1`
`  v.clear();`	`D1 D0`
`  v.emplace_back(2);`	`C2`
`  v.emplace_back(3);`	`C3`
`  v.front() = C(4);`	`C4 =4 D4`
`  v.pop_back();`	`D3`
`}`	`D4`