**ЛАБОРАТОРНА РОБОТА № 13**

**Варіант 2**

Сортування одновимірного масиву

***Мета*** : набуття практичних навиків роботи з одновимірними масивами, а саме сортування елементів масиву різними методами. Здійснення порівняння та аналізу ефективності використовуваних методів сортування.

**Хід роботи:**

Зміст

[Завдання 1 1](#_Toc86528721)

[Сортування обміном 2](#_Toc86528722)

[Сортування методом вибору 2](#_Toc86528723)

[Сортування вставками 3](#_Toc86528724)

[Сортування методом Шелла 4](#_Toc86528725)

[Самостійне завдання: 5](#_Toc86528726)

[№2 5](#_Toc86528727)

[Сортування обміном 5](#_Toc86528728)

[Методом вибору 6](#_Toc86528729)

[Сортування вставками 7](#_Toc86528730)

[Сортування методом Шелла 8](#_Toc86528731)

[№3 9](#_Toc86528732)

[Лістинінг пірамідального сортування (№4) 25](#_Toc86528733)

[№5 27](#_Toc86528734)

# 

# Завдання 1

1. Згенеруйте одновимірний масив і виведіть його на екран.

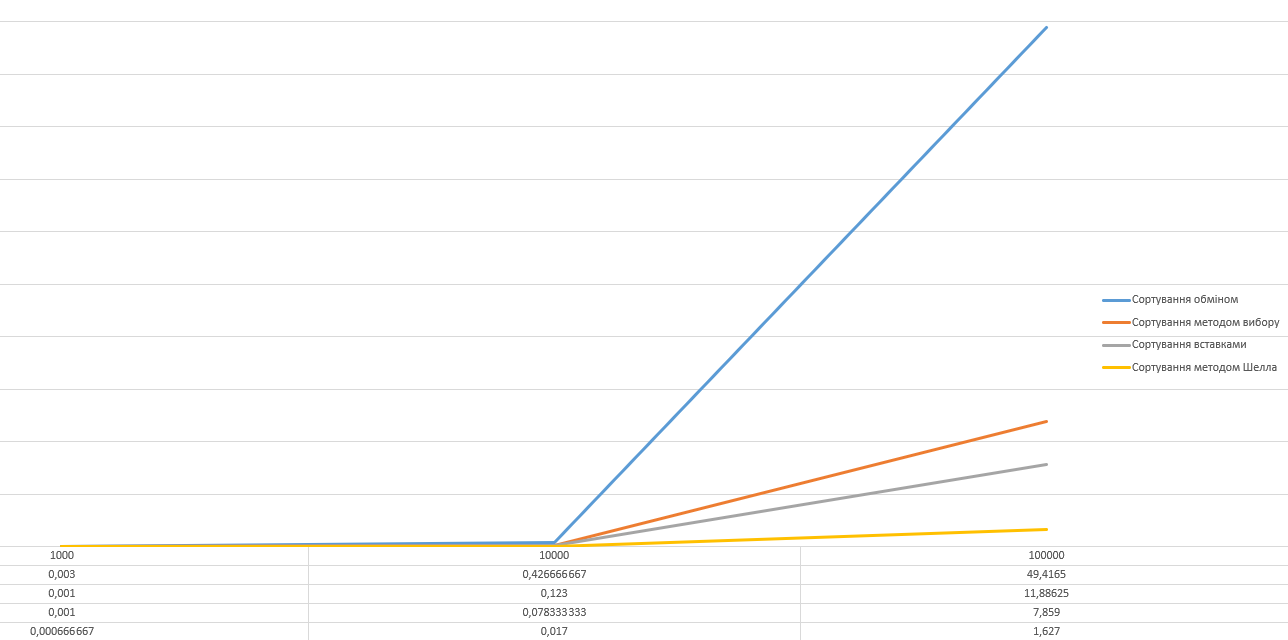
2. Відсортуйте масив кожним описаним методом.

3. Порівняйте швидкодію метода сортування при різних значеннях n (кількість елементів масиву) маленьких, середніх, великих. Доведіть розмір масиву до 100000 і більше елементів.

4. Запускаючи програму не менше десяти разів для кожного методу, отримати час t сортування. Побудувати залежність t=f(n) на одному графіку для різних методів сортування (у табличному і графічному виглядах).

5. Які з вивчених методів сортування дають найкращий результат для великих масивів? Який метод найгірший?

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод | n = | 1000 | 1000 | 1000 | 10000 | 10000 | 10000 | 100000 | 100000 | 100000 | 100000 |
| обміном | | 0.003 | 0.003 | 0.003 | 0.424 | 0.43 | 0.426 | 49.296 | 50.06 | 49.224 | 49.086 |
| методом вибору | | 0.001 | 0.001 | 0.001 | 0.118 | 0.133 | 0.118 | 11.812 | 11.969 | 11.859 | 11.905 |
| вставками | | 0.001 | 0.001 | 0.001 | 0.078 | 0.08 | 0.077 | 7.852 | 7.928 | 7.837 | 7.819 |
| методом Шелла | | 0.001 | 0 | 0.001 | 0.017 | 0.017 | 0.017 | 1.606 | 1.655 | 1.612 | 1.635 |

****

Сортування методом Шелла дало накращий результат

Найгірший методом було сортування обміном

## Сортування обміном

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, a[100000], i, fl, c;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

a[i] = 0 + rand() % 101;

}

clock\_t begin = clock();

do

{

fl = 0;

for (int i = 1; i <= x; i++)

if (a[i - 1] > a[i])

{

c = a[i];

a[i] = a[i - 1];

a[i - 1] = c;

fl = 1;

}

} while (fl);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf", time\_spent);

for (i = 1; x >= i; i++) {

}

}

return 0;

}

## Сортування методом вибору

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, imin, a;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

ar[i] = 0 + rand() % 101;

}

clock\_t begin = clock();

for (int i = 1; i <= x - 1; i++)

{

imin = i;

for (int j = i + 1; j <= x; j++)

if (ar[j] < ar[imin]) imin = j;

a = ar[i];

ar[i] = ar[imin];

ar[imin] = a;

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf", time\_spent);

for (i = 1; x >= i; i++) {

}

}

return 0;

}

## Сортування вставками

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, c;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

ar[i] = 0 + rand() % 101;

}

clock\_t begin = clock();

for (int i = 1; i <= x; i++)

{

c = ar[i];

for (int j = i - 1; j >= 0 && ar[j] > c; j--)

{

ar[j + 1] = ar[j];

ar[j] = c;

}

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf", time\_spent);

for (i = 1; x >= i; i++) {

}

}

return 0;

}

## Сортування методом Шелла

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, c;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

ar[i] = 0 + rand() % 101;

}

clock\_t begin = clock();

int step = x / 2;

while (step > 0)

{

for (int i = 1; i <= (x - step); i++)

{

int j = i;

while (j >= 0 && ar[j] > ar[j + step])

{

int c = ar[j];

ar[j] = ar[j + step];

ar[j + step] = c;

j--;

}

}

step = step / 2;

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf", time\_spent);

for (i = 1; x >= i; i++) {

}

}

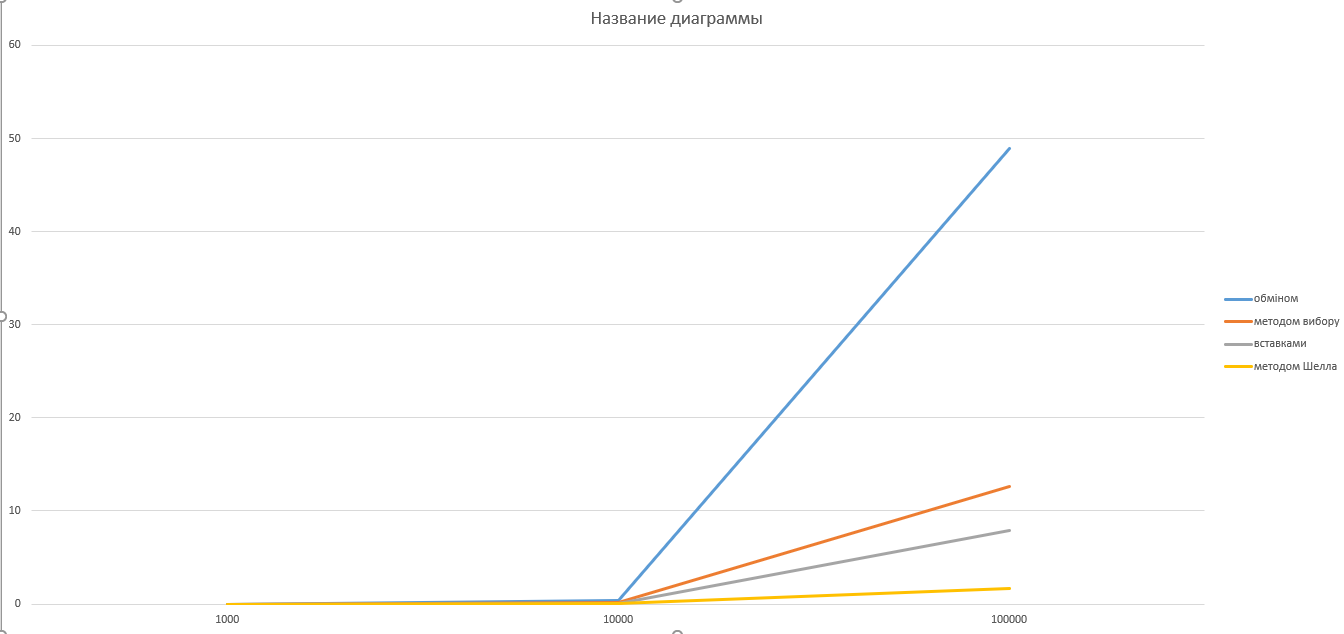
return 0;

}

# Самостійне завдання:

# №2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод | n = | 1000 | 1000 | 1000 | 10000 | 10000 | 10000 | 100000 | 100000 | 100000 | 100000 |
| обміном | | 0.003 | 0.003 | 0.003 | 0.431 | 0.428 | 0.429 | 49.008 | 49.029 | 48.984 | 48.913 |
| методом вибору | | 0,001 | 0,002 | 0,001 | 0.129 | 0.127 | 0.125 | 12.737 | 12.706 | 12.606 | 12.661 |
| вставками | | 0.001 | 0.001 | 0.001 | 0.078 | 0.076 | 0.077 | 7.847 | 7.856 | 7.864 | 7.826 |
| методом Шелла | | 0.001 | 0 | 0 | 0.017 | 0.017 | 0.017 | 1.63 | 1.639 | 1.631 | 1.653 |

****

## Сортування обміном

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, a[100000], i, fl, c;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

a[i] = 0 + rand() % 101;

//printf("%d ", a[i]);

}

clock\_t begin = clock();

do

{

fl = 0;

for (int i = 1; i <= x; i++)

if (a[i + 1] > a[i])

{

c = a[i];

a[i] = a[i + 1];

a[i + 1] = c;

fl = 1;

}

} while (fl);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf\n", time\_spent);

for (i = 1; x >= i; i++) {

//printf("%d ", a[i]);

}

}

return 0;

}

## Методом вибору

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, fl, c, imin, a;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 0; x >= i; i++) {

ar[i] = 0 + rand() % 101;

//printf("%d ", ar[i]);

}

clock\_t begin = clock();

for (int i = 1; i < x - 1; i++)

{

imin = i;

for (int j = i + 1; j <= x; j++)

if (ar[j] >= ar[imin]) imin = j;

a = ar[i];

ar[i] = ar[imin];

ar[imin] = a;

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf\n", time\_spent);

for (i = 1; x >= i; i++) {

//printf("%d ", ar[i]);

}

}

return 0;

}

## Сортування вставками

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, fl, c, imin, a;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

ar[i] = 0 + rand() % 101;

//printf("%d ", ar[i]);

}

clock\_t begin = clock();

for (int i = 1; i <= x; i++)

{

c = ar[i];

for (int j = i - 1; j > 0 && ar[j] > c; j--)

{

ar[j + 1] = ar[j];

ar[j] = c;

}

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf\n", time\_spent);

for (i = 1; x >= i; i++) {

//printf("%d ", ar[i]);

}

}

return 0;

}

## Сортування методом Шелла

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int x, ar[100000], i, fl, c, imin, a;

for (;;) {

printf("\nn = ");

scanf\_s("%d", &x);

for (i = 1; x >= i; i++) {

ar[i] = 0 + rand() % 101;

//printf("%d ", ar[i]);

}

clock\_t begin = clock();

int step = x / 2;

while (step > 0)

{

for (int i = 1; i <= (x - step); i++)

{

int j = i;

while (j > 0 && ar[j] < ar[j + step])

{

int c = ar[j];

ar[j] = ar[j + step];

ar[j + step] = c;

j--;

}

}

step = step / 2;

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("\n%lf\n", time\_spent);

for (i = 1; x >= i; i++) {

//printf("%d ", ar[i]);

}

}

return 0;

}

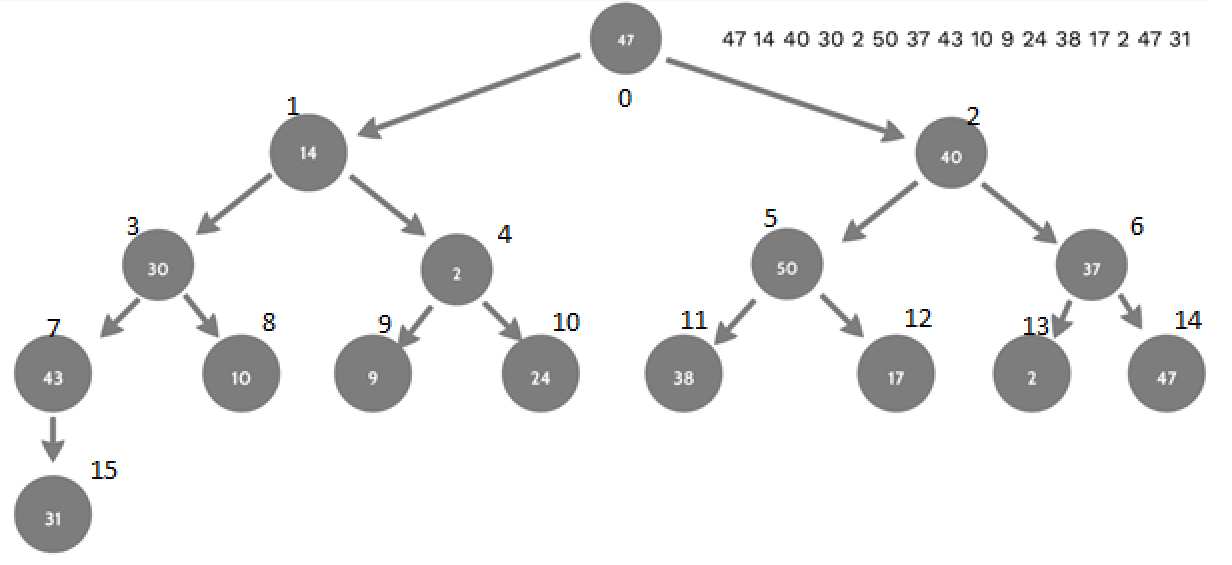
# №3

\*Дано масив із 16 чисел.

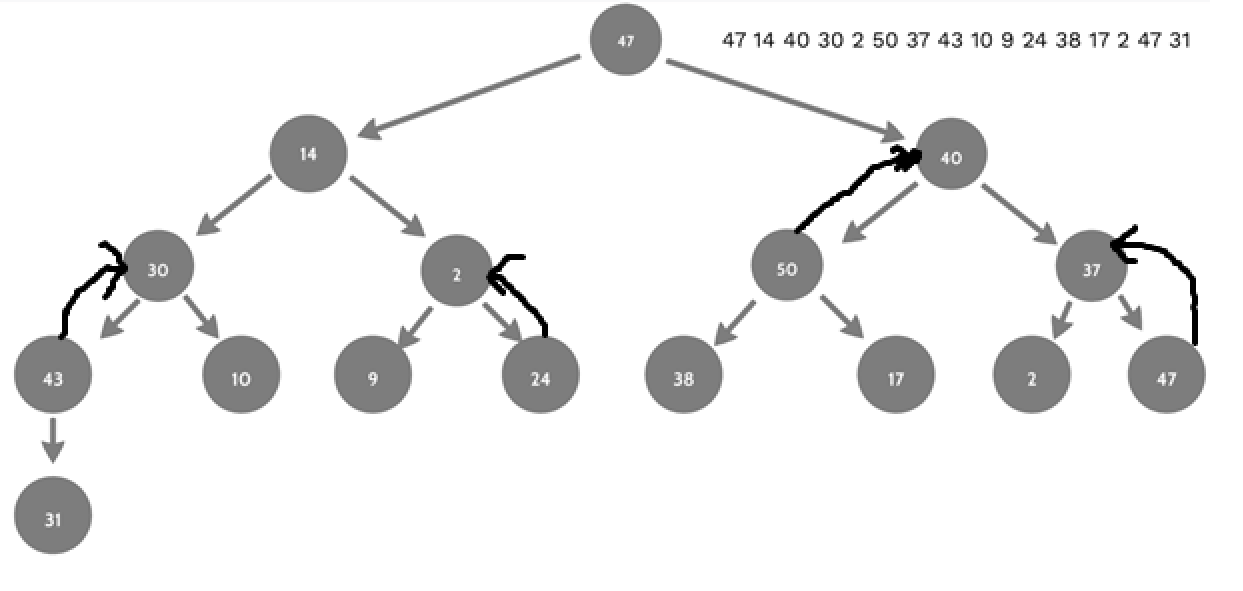
• Розписати покроково результат пірамідального сортування і подати у вигляді таблиці.

• Побудувати дерево.

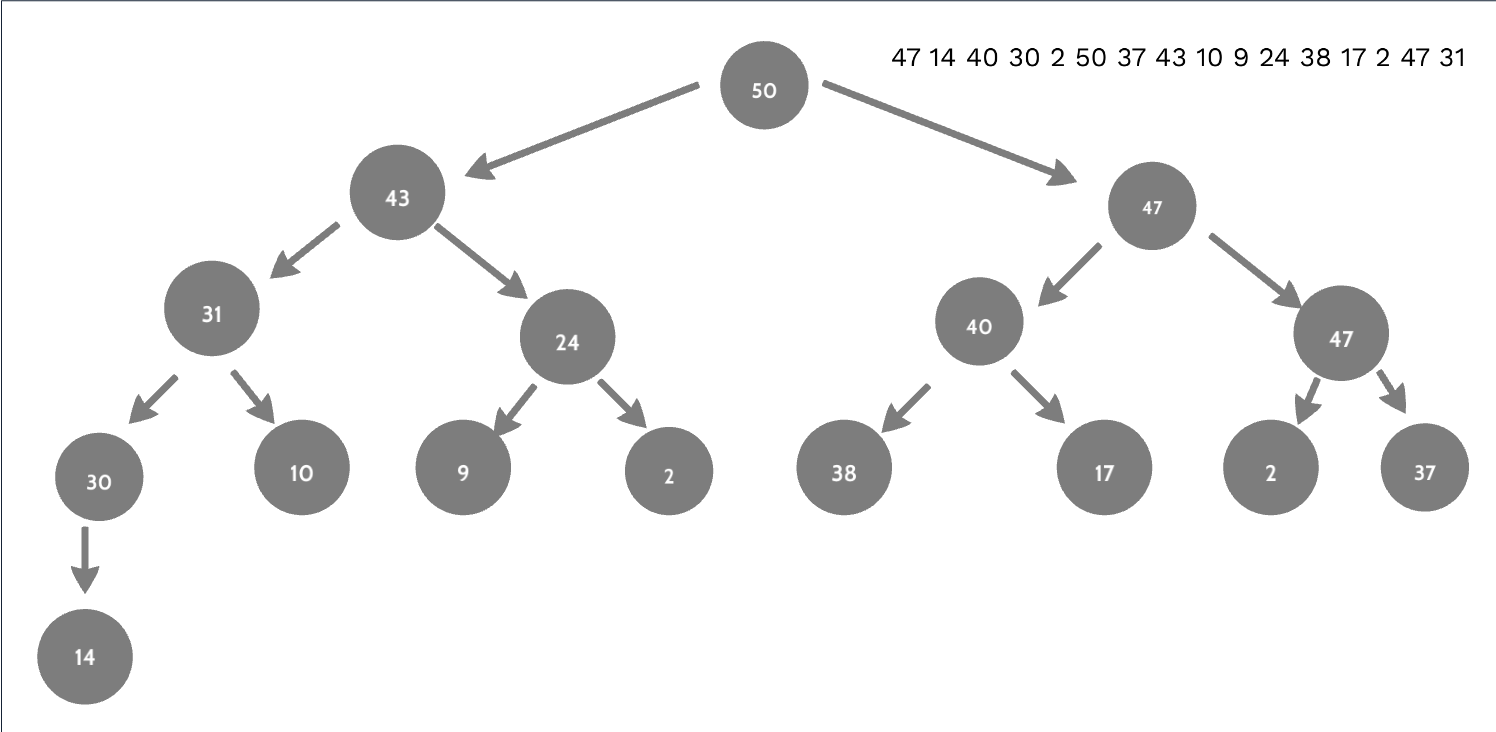
Відрахунок масива починається ось так



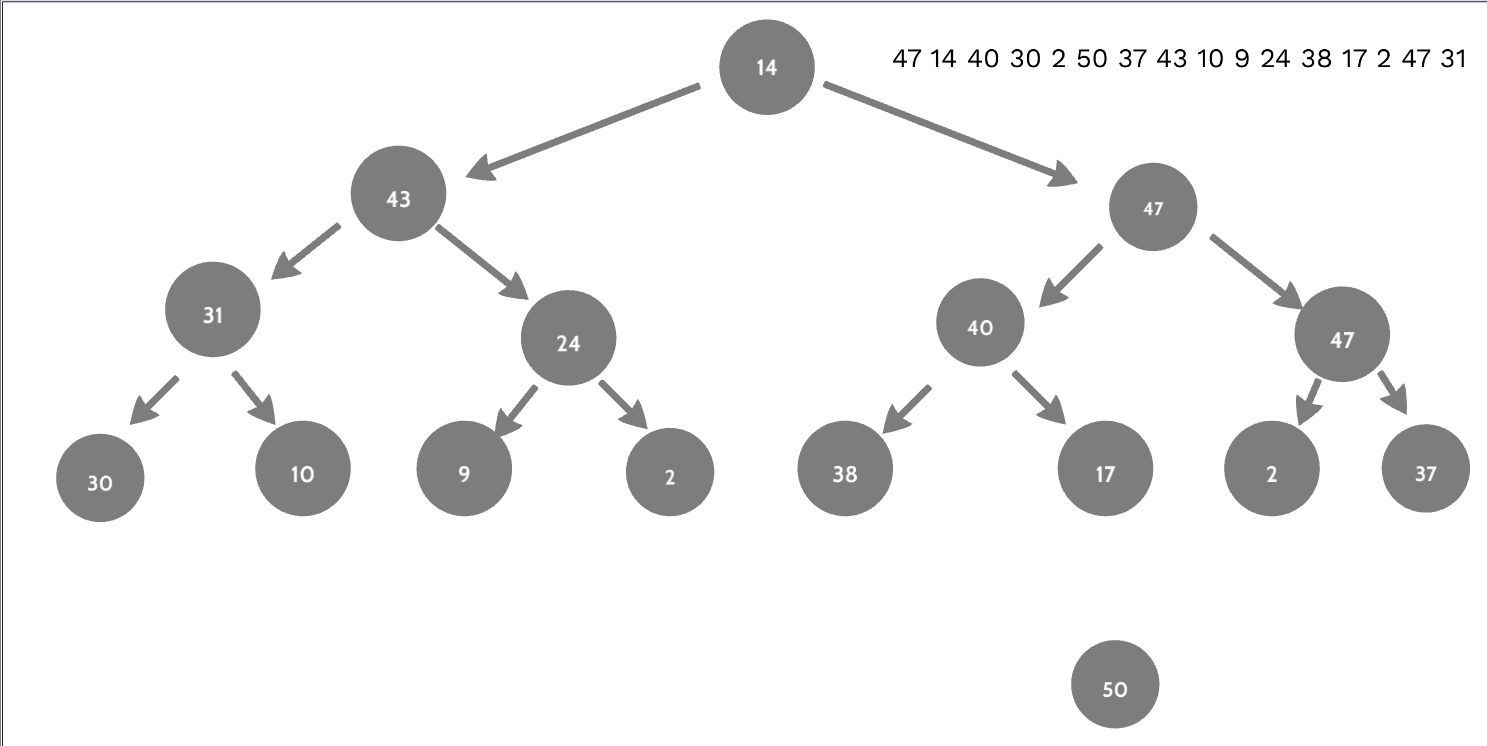
Далі я кожне число яке більше, ніж те що зверху, замінював місцями, наприклад

****

І так кілька разів, допоки найбільші числа, не будуть зверху

****

Далі я замінив нульовий та останій елемент масиву, відділивши при цьому вже новий шістнадцятий елемент масиву

****

Цей “відрубаний” елемент і є останім елементом відсортованого масиву

Ми повторюємо ці операції доти, доки в нас не закінчаться елменти в цьому дереві.

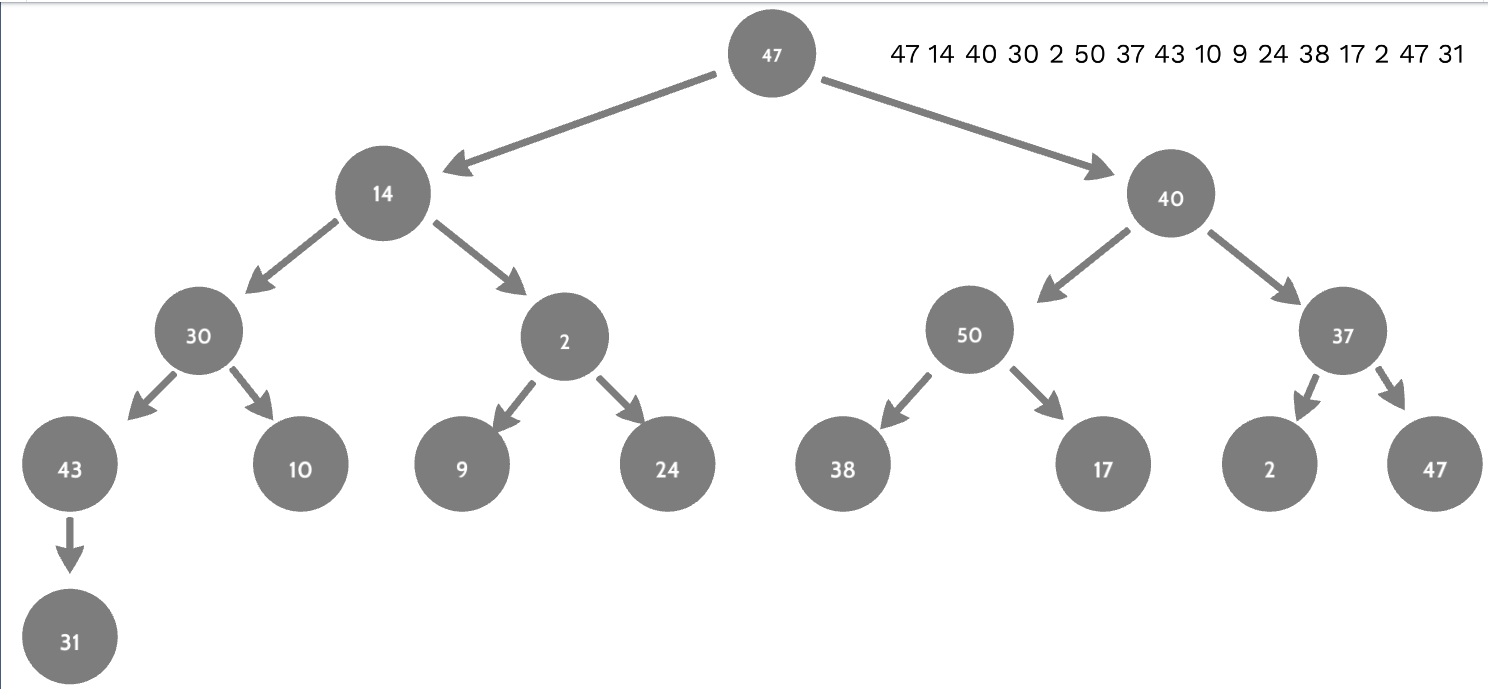
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **47** | **14** | **40** | **30** | **2** | **50** | **37** | **43** | **10** | **9** | **24** | **38** | **17** | **2** | **47** | **31** |
| **14** | **43** | **47** | **31** | **24** | **40** | **47** | **30** | **10** | **9** | **2** | **38** | **17** | **2** | **37** | **50** |
| **14** | **43** | **47** | **31** | **24** | **40** | **37** | **30** | **10** | **9** | **2** | **38** | **17** | **2** | **47** | **50** |
| **2** | **43** | **40** | **31** | **24** | **38** | **37** | **30** | **10** | **9** | **2** | **14** | **17** | **47** | **47** | **50** |
| **17** | **31** | **40** | **30** | **24** | **38** | **37** | **2** | **10** | **9** | **2** | **14** | **43** | **47** | **47** | **50** |
| **14** | **31** | **38** | **30** | **24** | **17** | **37** | **2** | **10** | **9** | **2** | **40** | **43** | **47** | **47** | **50** |
| **2** | **31** | **37** | **30** | **24** | **17** | **14** | **2** | **10** | **9** | **38** | **40** | **43** | **47** | **47** | **50** |
| **9** | **31** | **17** | **30** | **24** | **2** | **14** | **2** | **10** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **9** | **30** | **17** | **10** | **24** | **2** | **14** | **2** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **2** | **24** | **17** | **10** | **9** | **2** | **14** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **14** | **10** | **17** | **2** | **9** | **2** | **24** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **2** | **10** | **14** | **2** | **9** | **17** | **24** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **9** | **10** | **2** | **2** | **14** | **17** | **24** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **2** | **9** | **2** | **10** | **14** | **17** | **24** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |
| **2** | **2** | **9** | **10** | **14** | **17** | **24** | **30** | **31** | **37** | **38** | **40** | **43** | **47** | **47** | **50** |

**----------------------------------------------------------------------------**

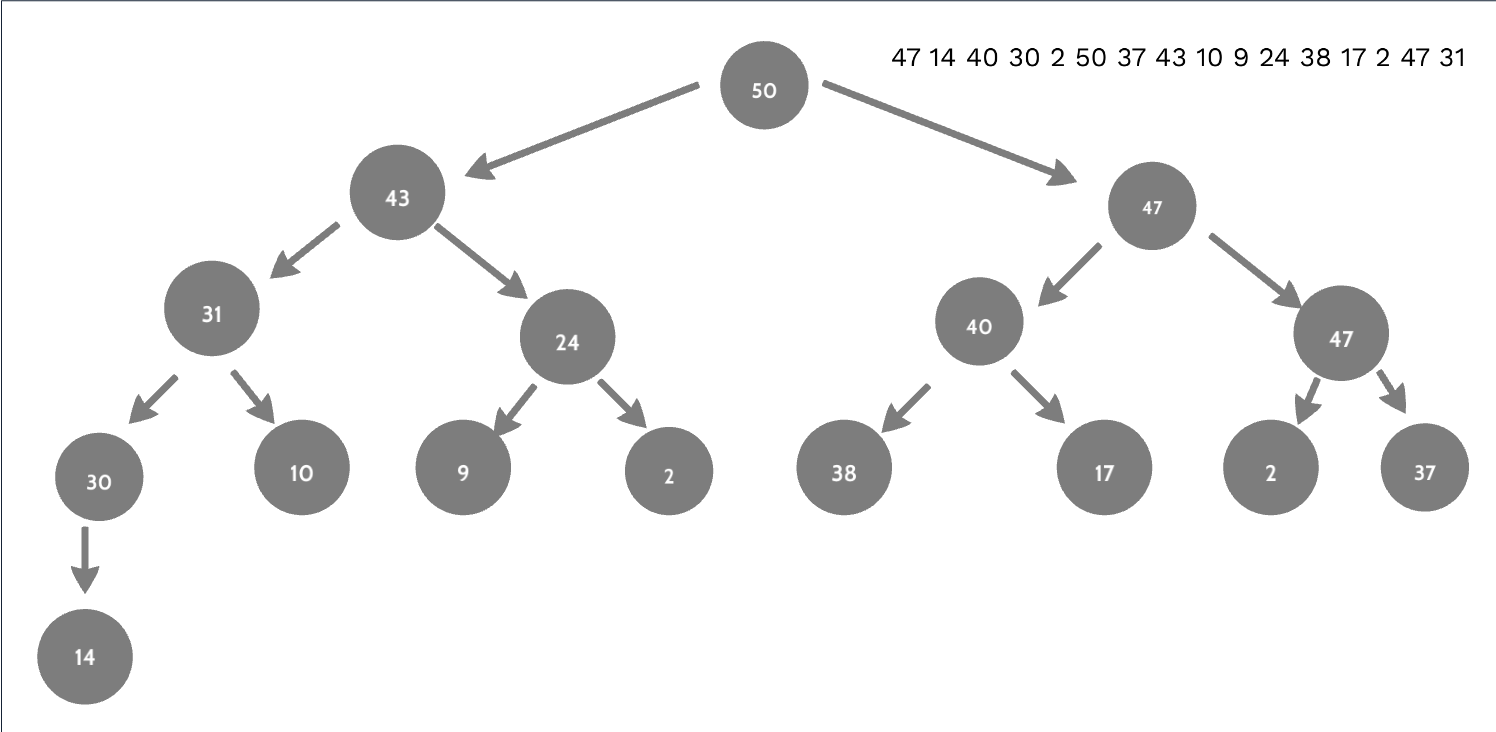
**----------------------------------------------------------------------------**

**----------------------------------------------------------------------------**

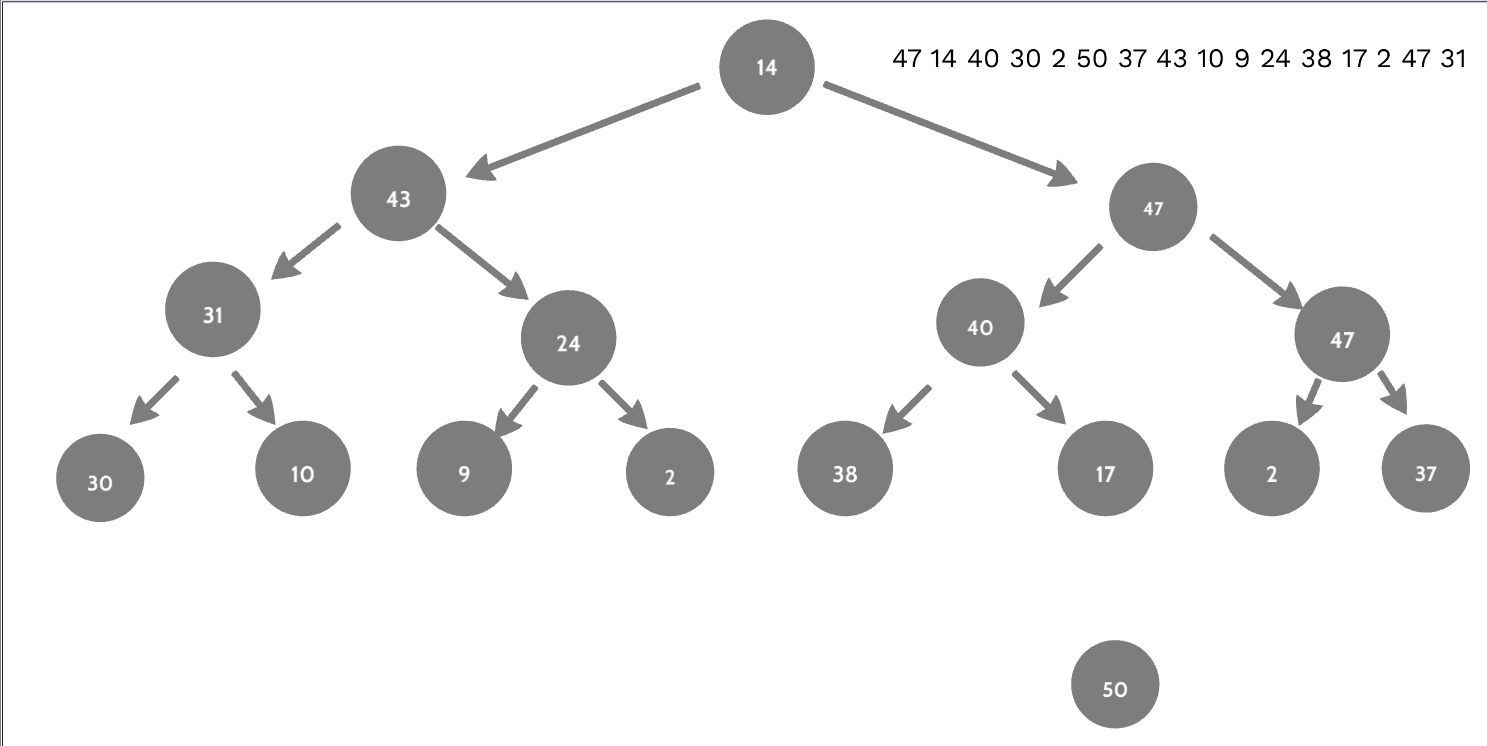
**----------------------------------------------------------------------------**

****

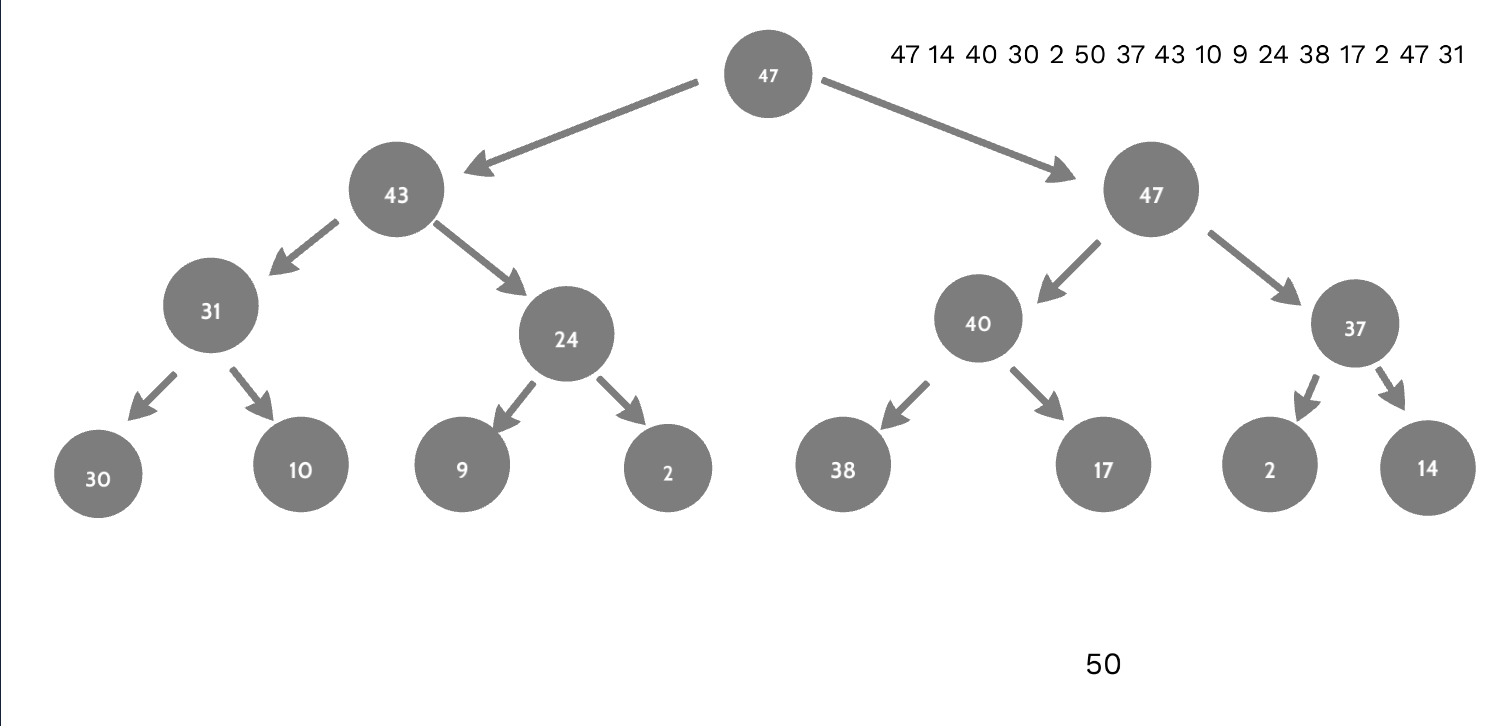
**----------------------------------------------------------------------------**

****

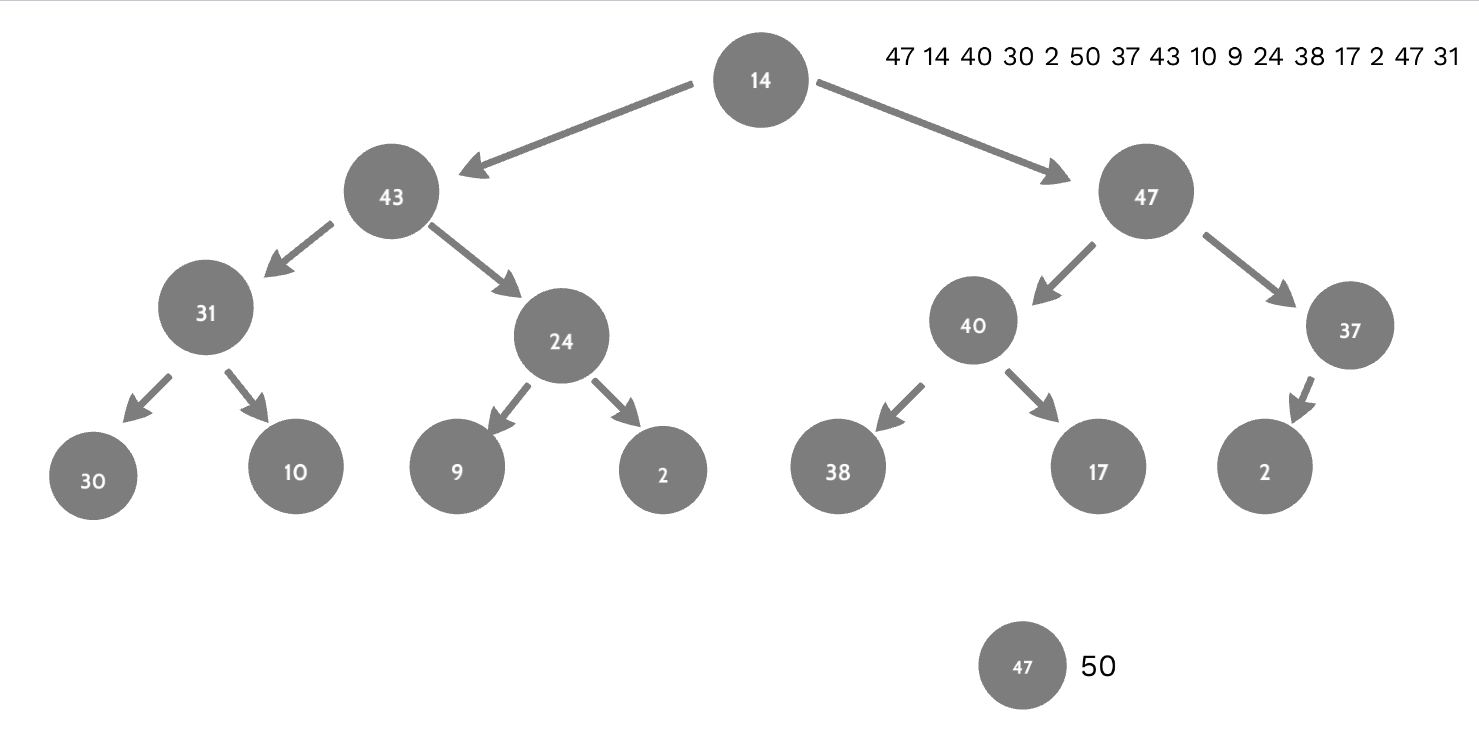
**----------------------------------------------------------------------------**

****

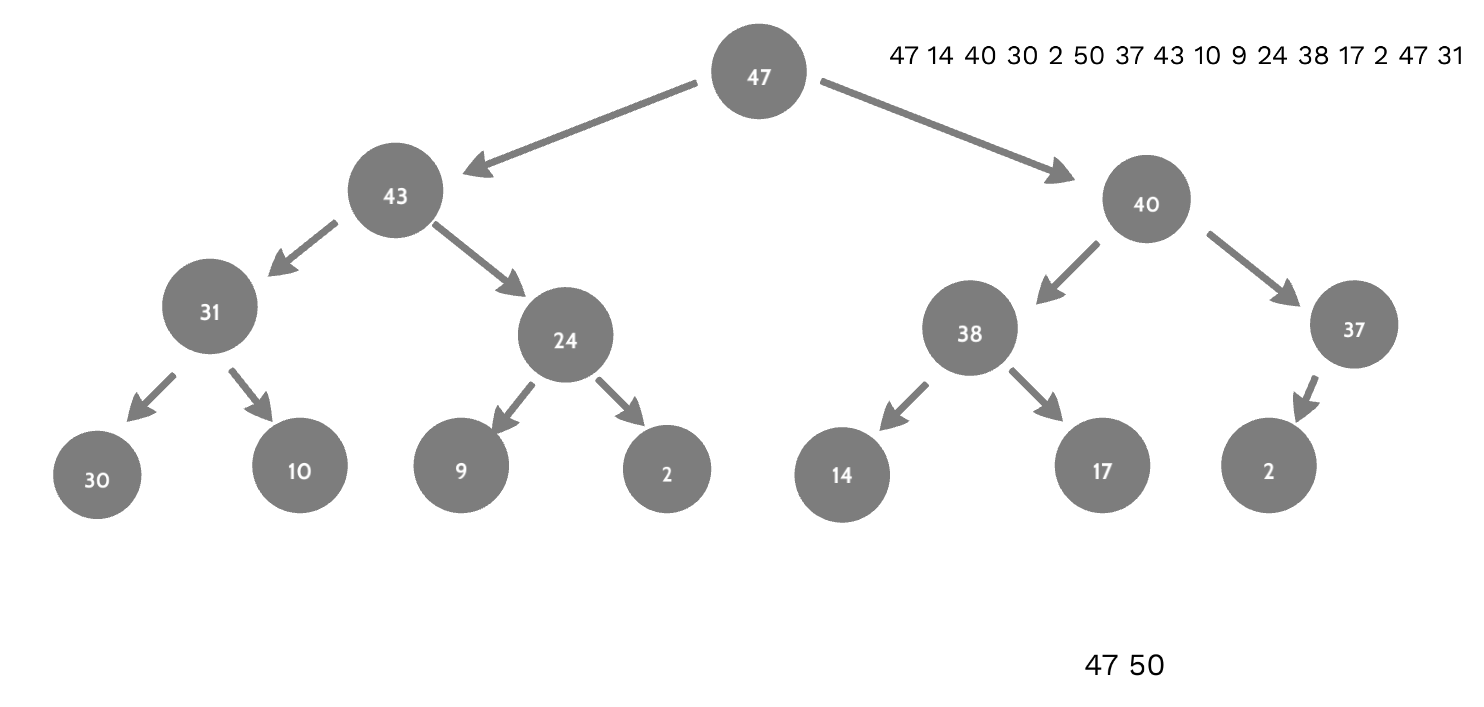
**----------------------------------------------------------------------------**

****

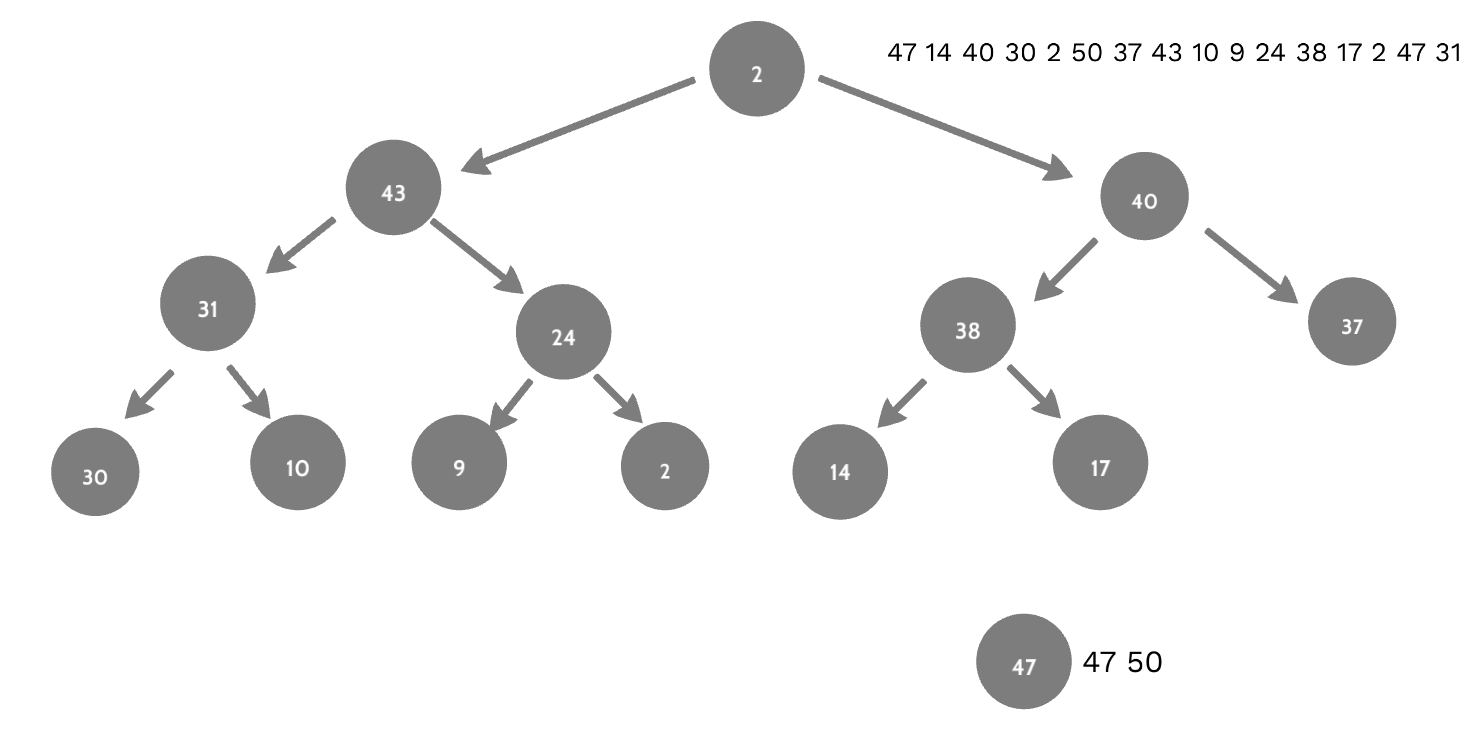
**----------------------------------------------------------------------------**

****

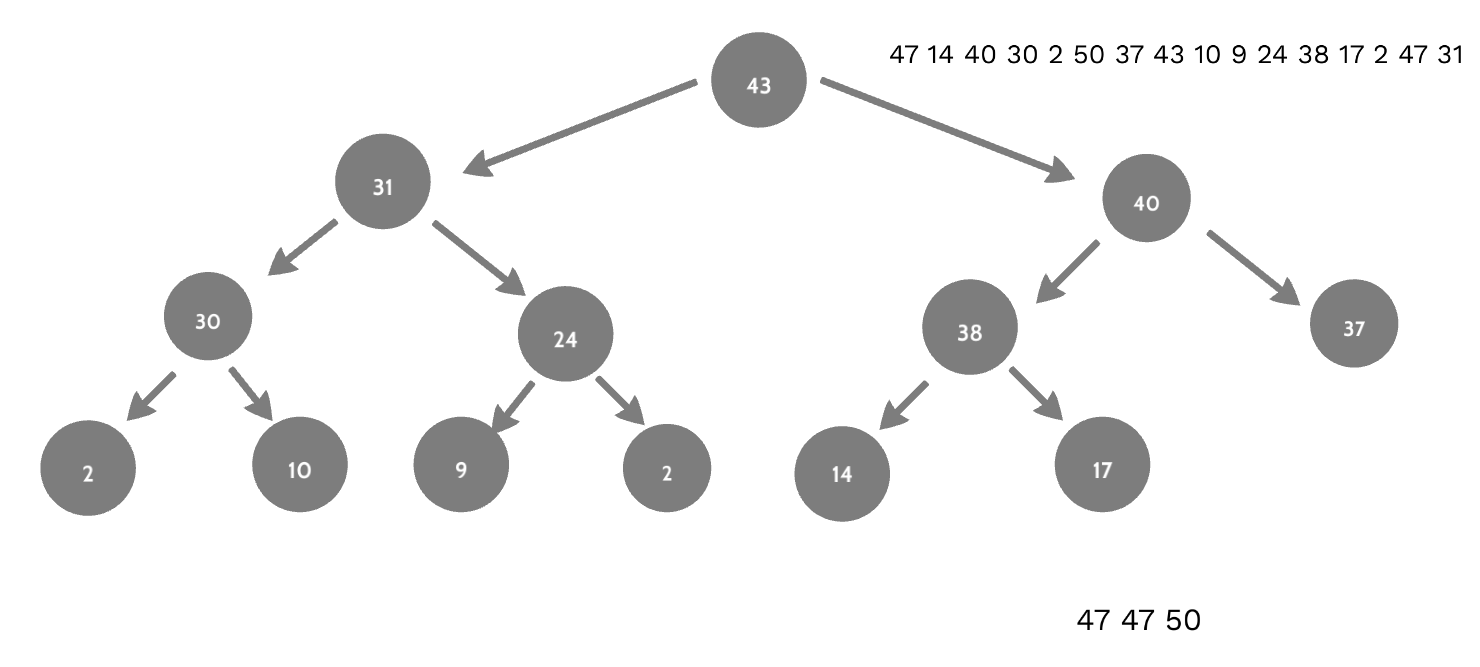
**----------------------------------------------------------------------------**

****

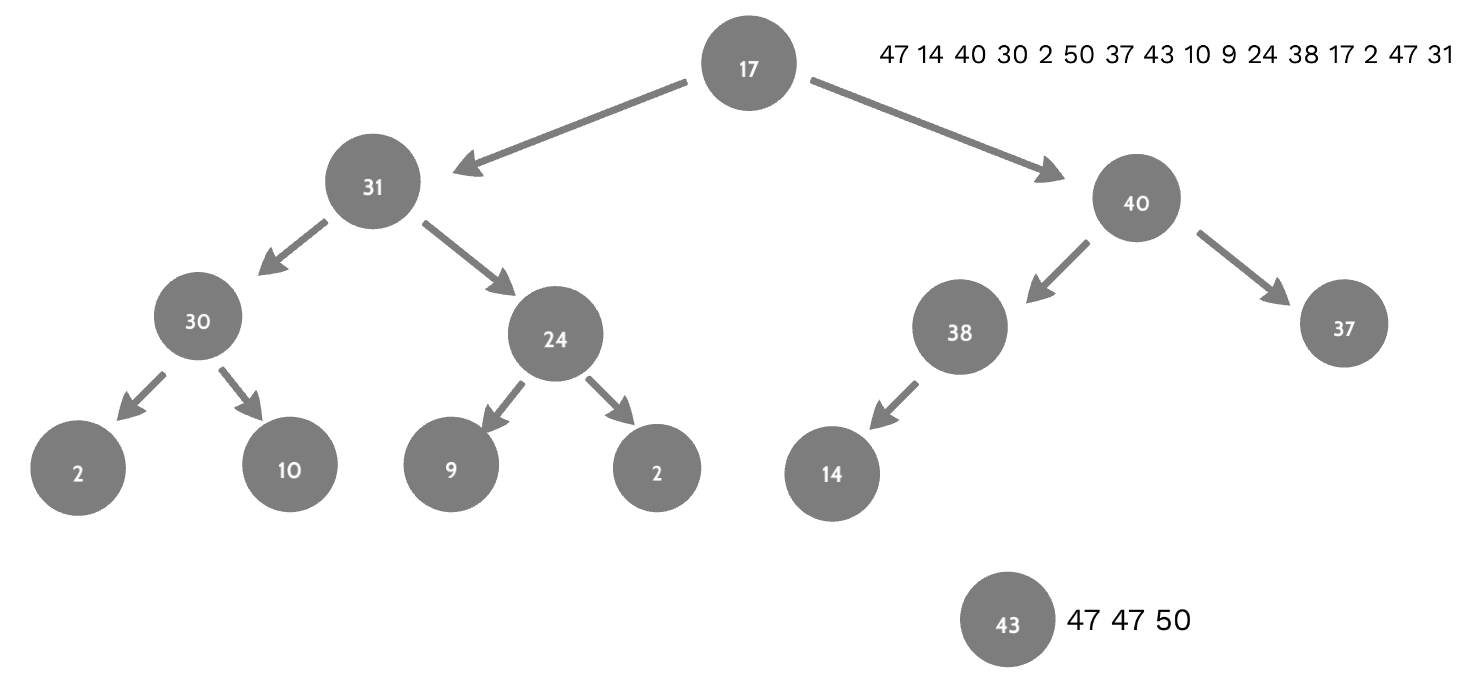
**----------------------------------------------------------------------------**

****

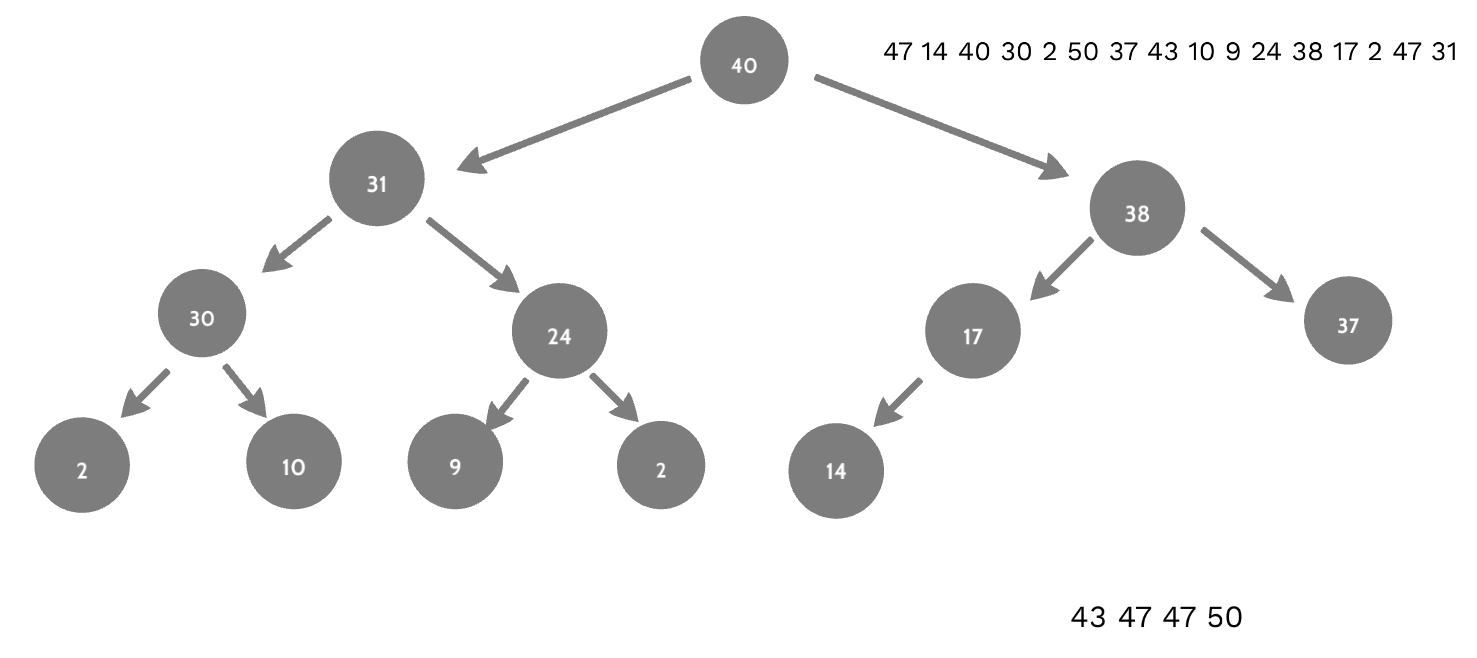
**----------------------------------------------------------------------------**

****

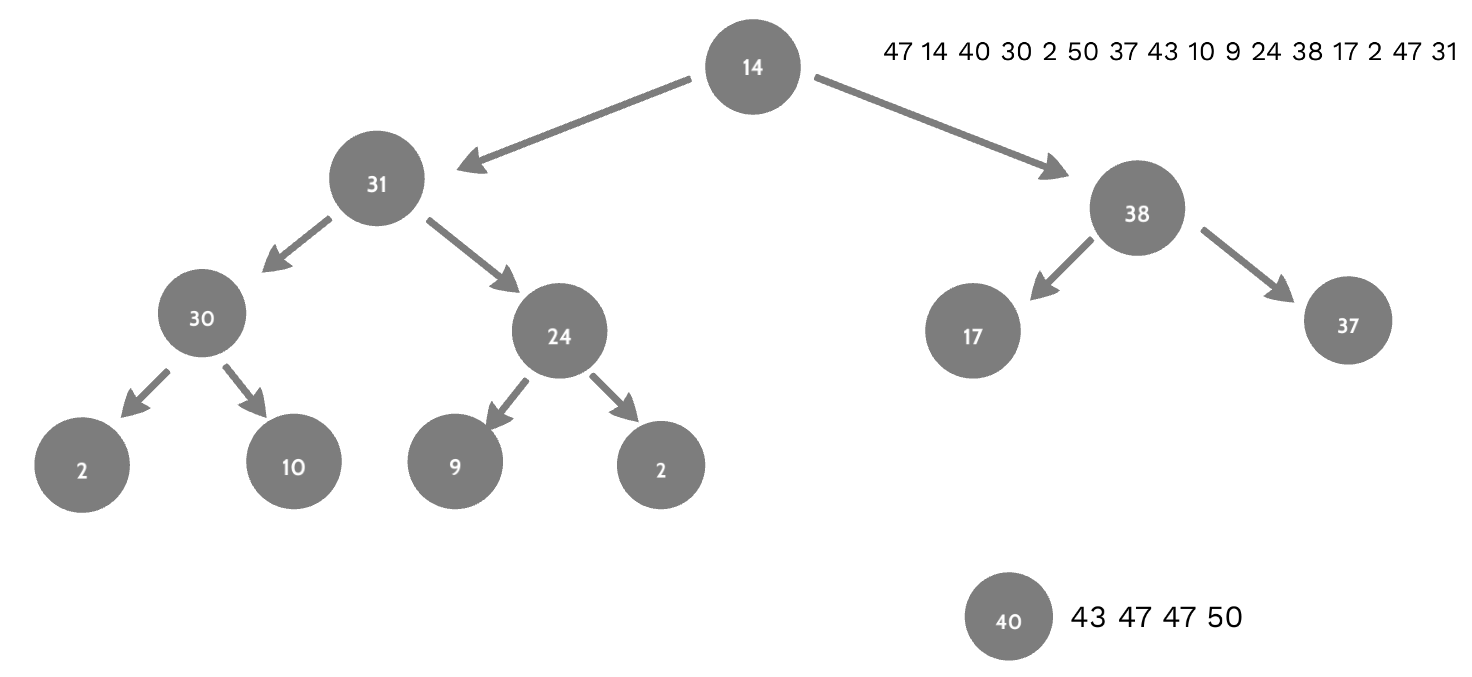
**----------------------------------------------------------------------------**

****

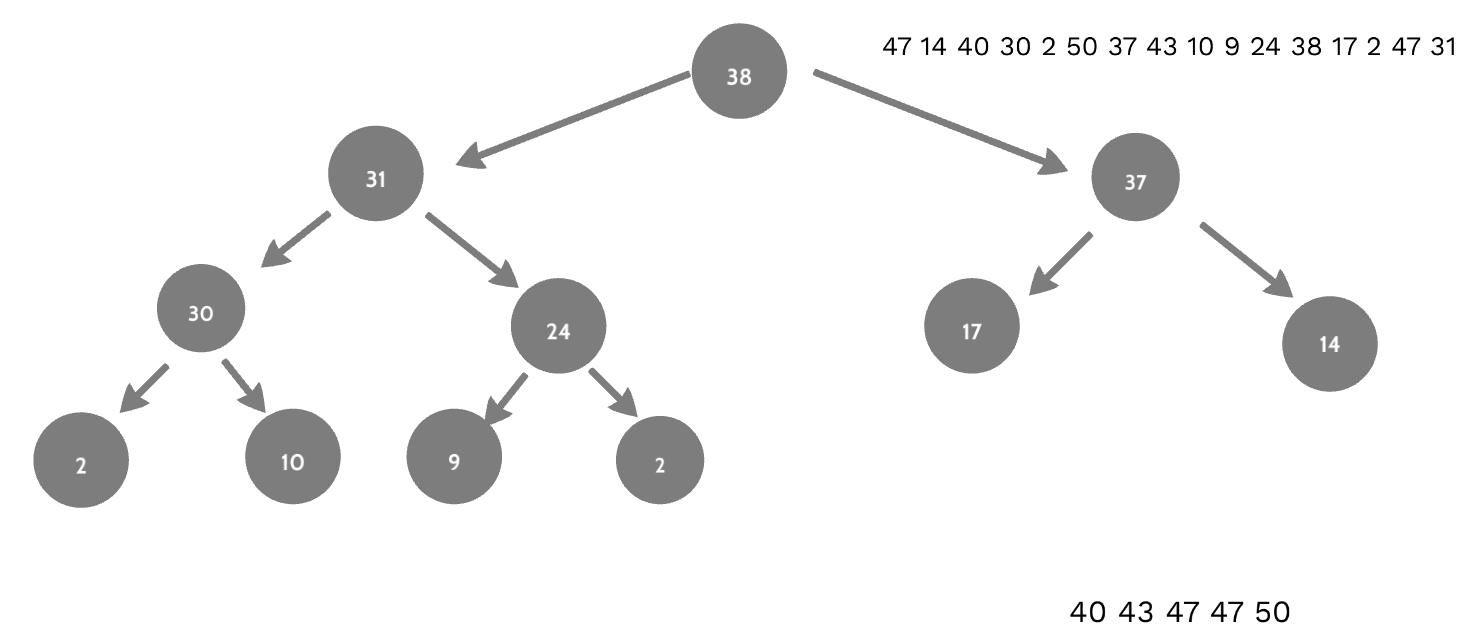
**----------------------------------------------------------------------------**

****

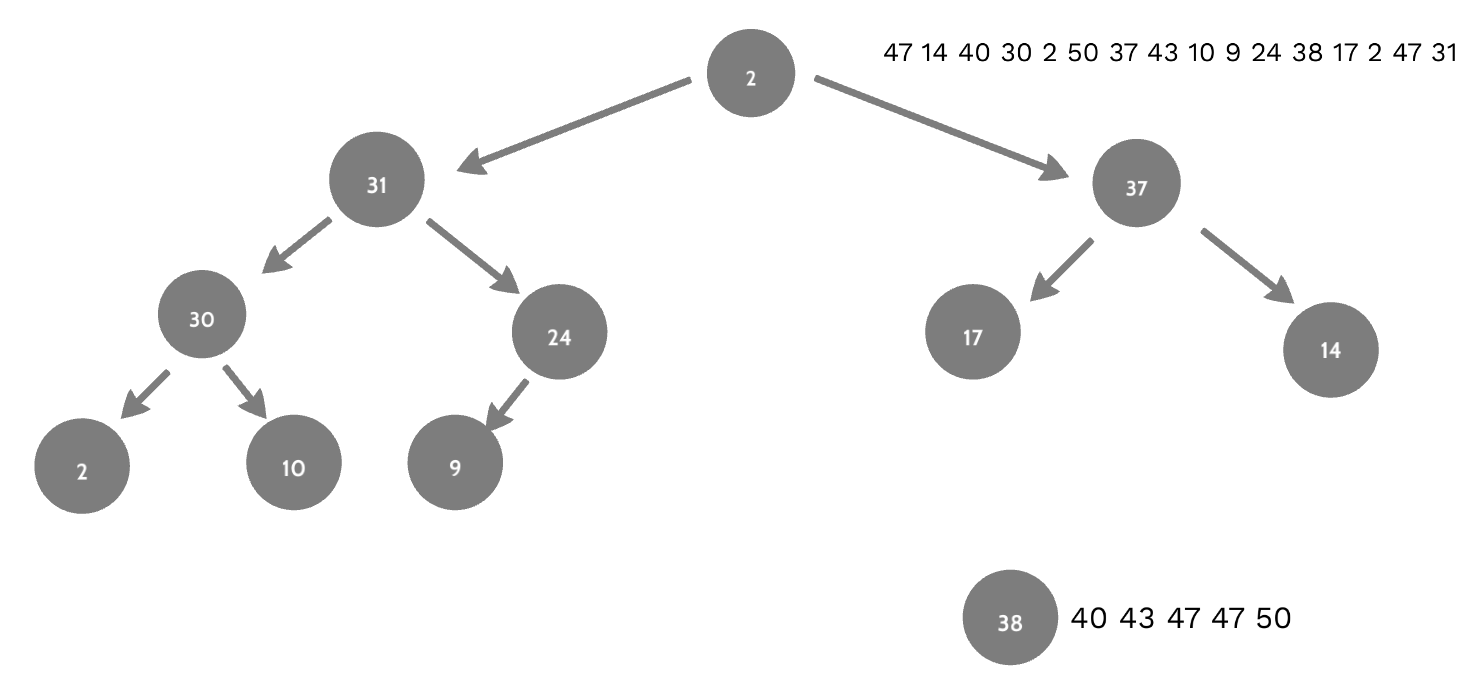
**----------------------------------------------------------------------------**

****

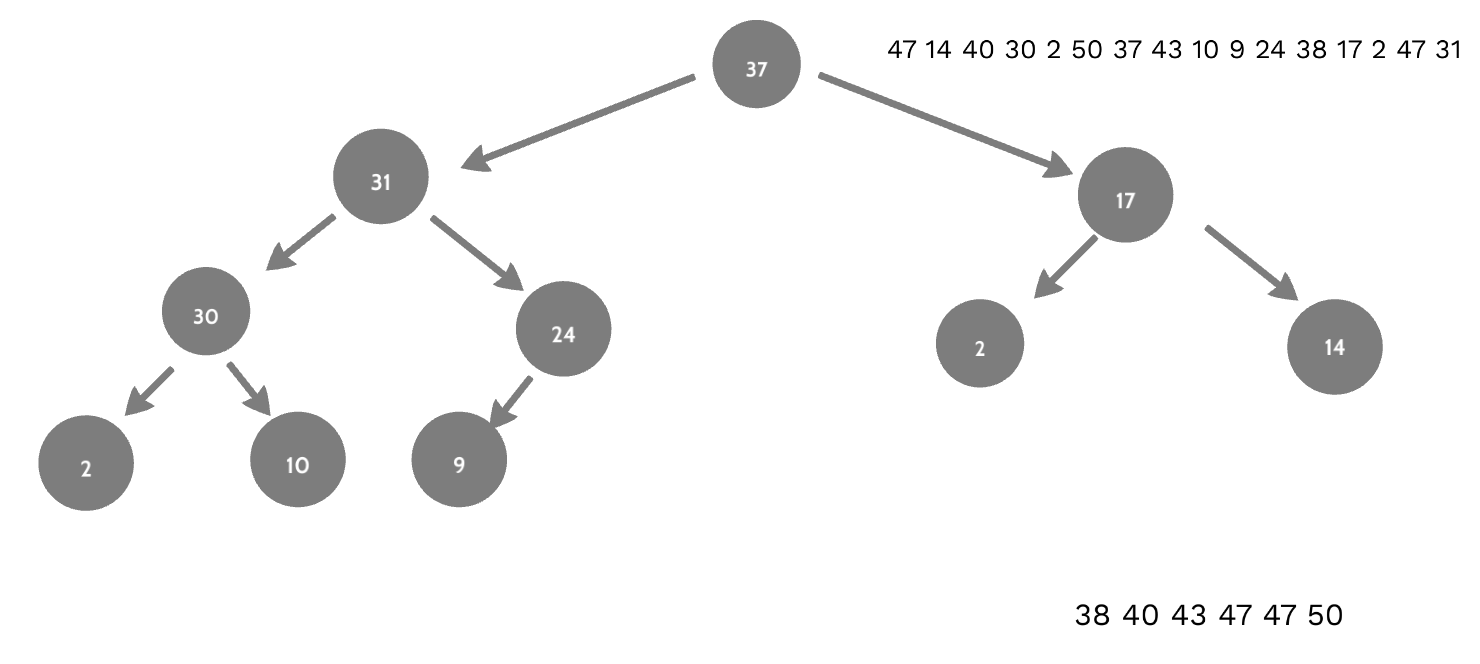
**----------------------------------------------------------------------------**

****

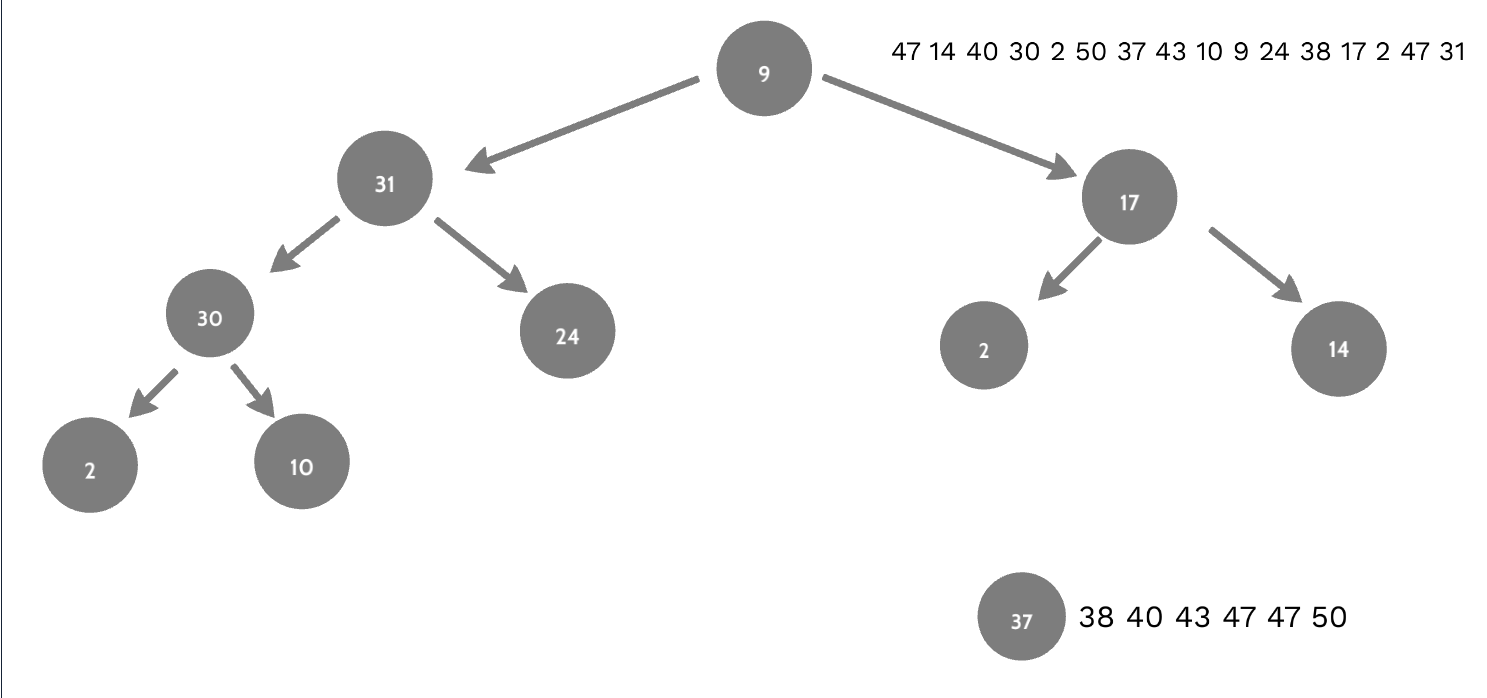
**----------------------------------------------------------------------------**

****

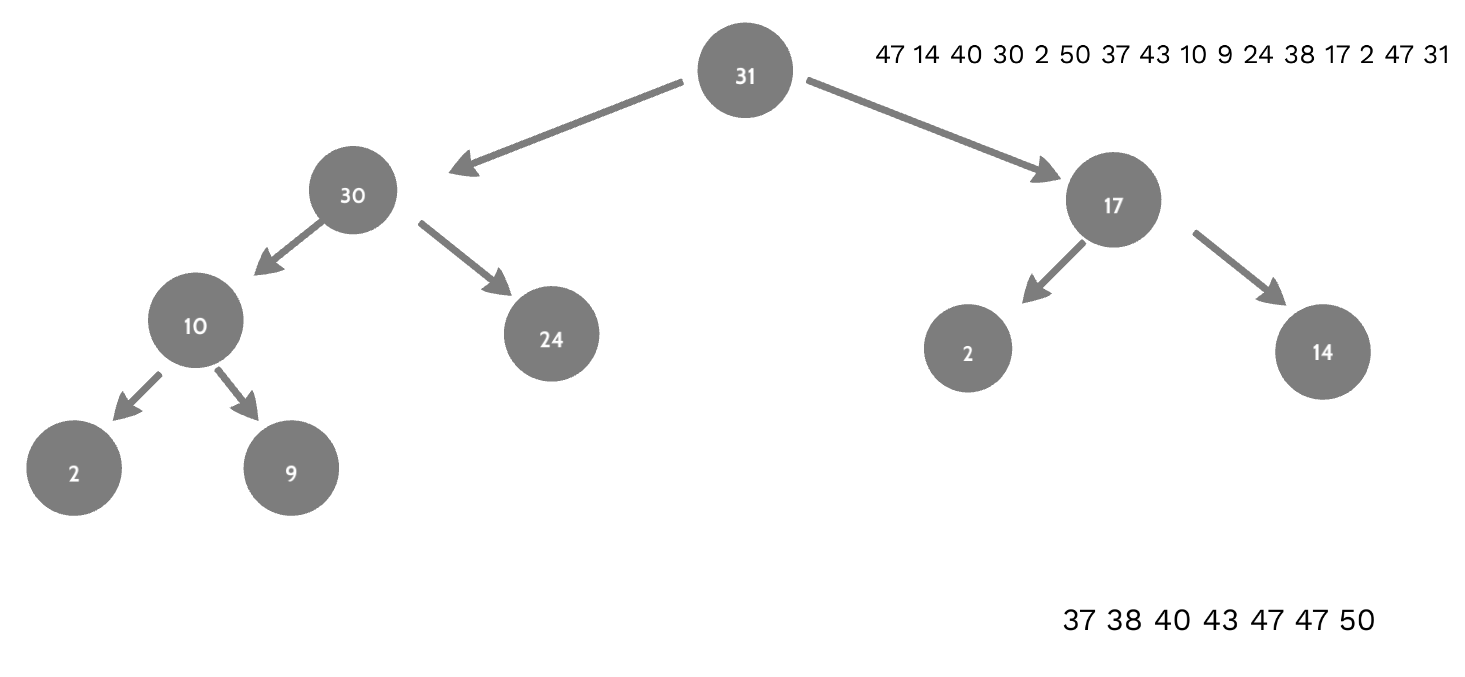
**----------------------------------------------------------------------------**

****

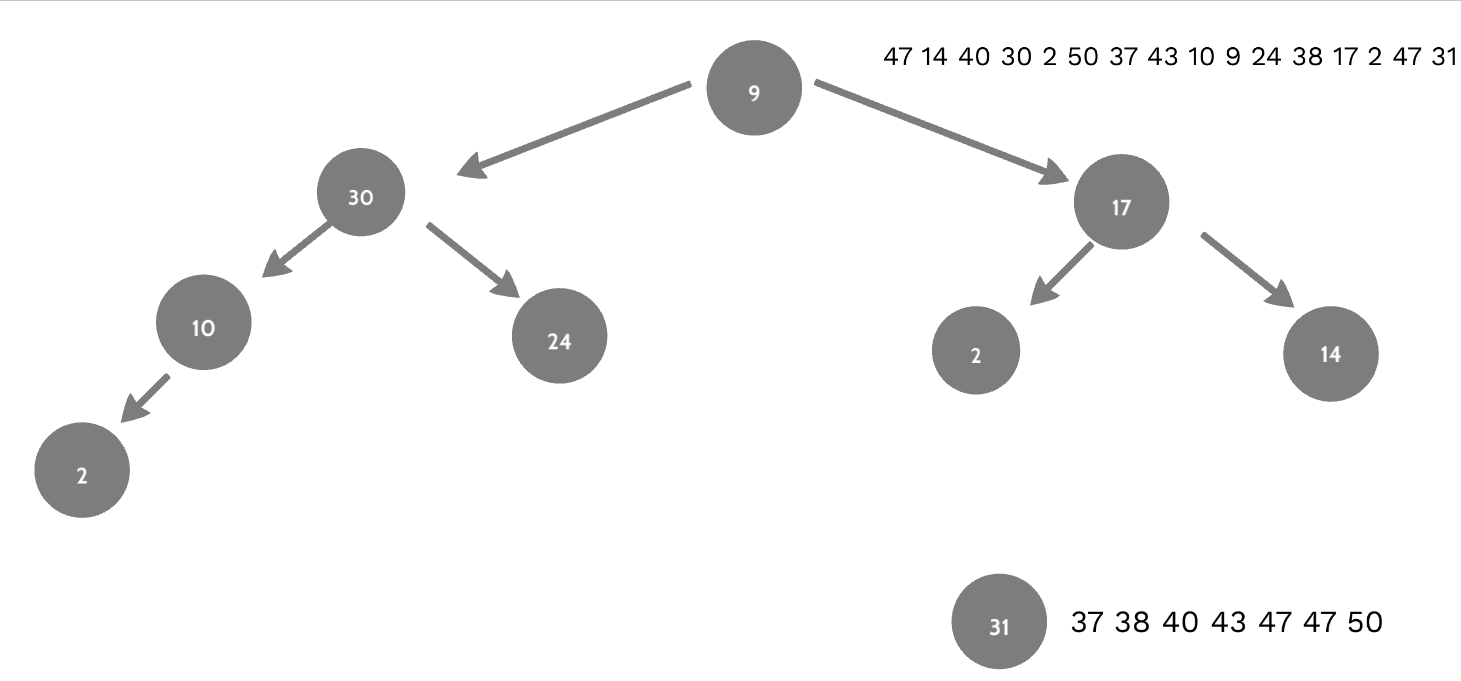
**----------------------------------------------------------------------------**

****

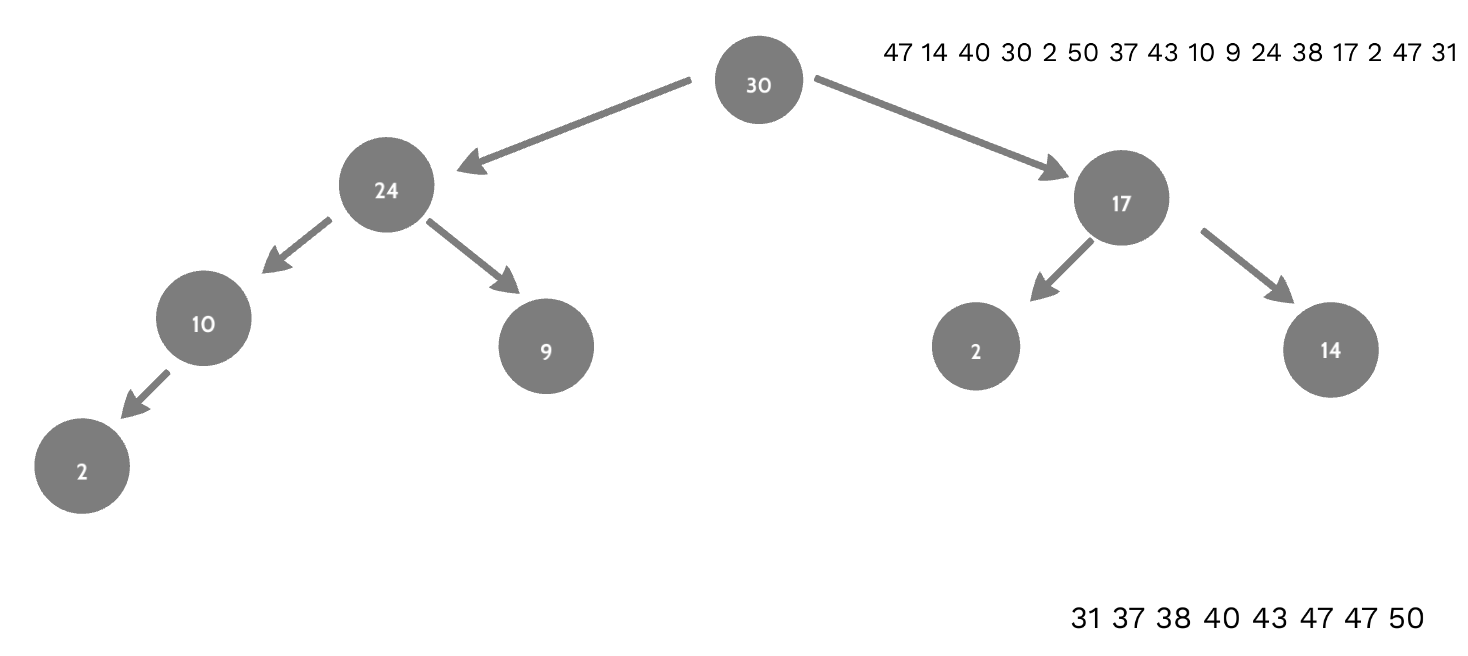
**----------------------------------------------------------------------------**

****

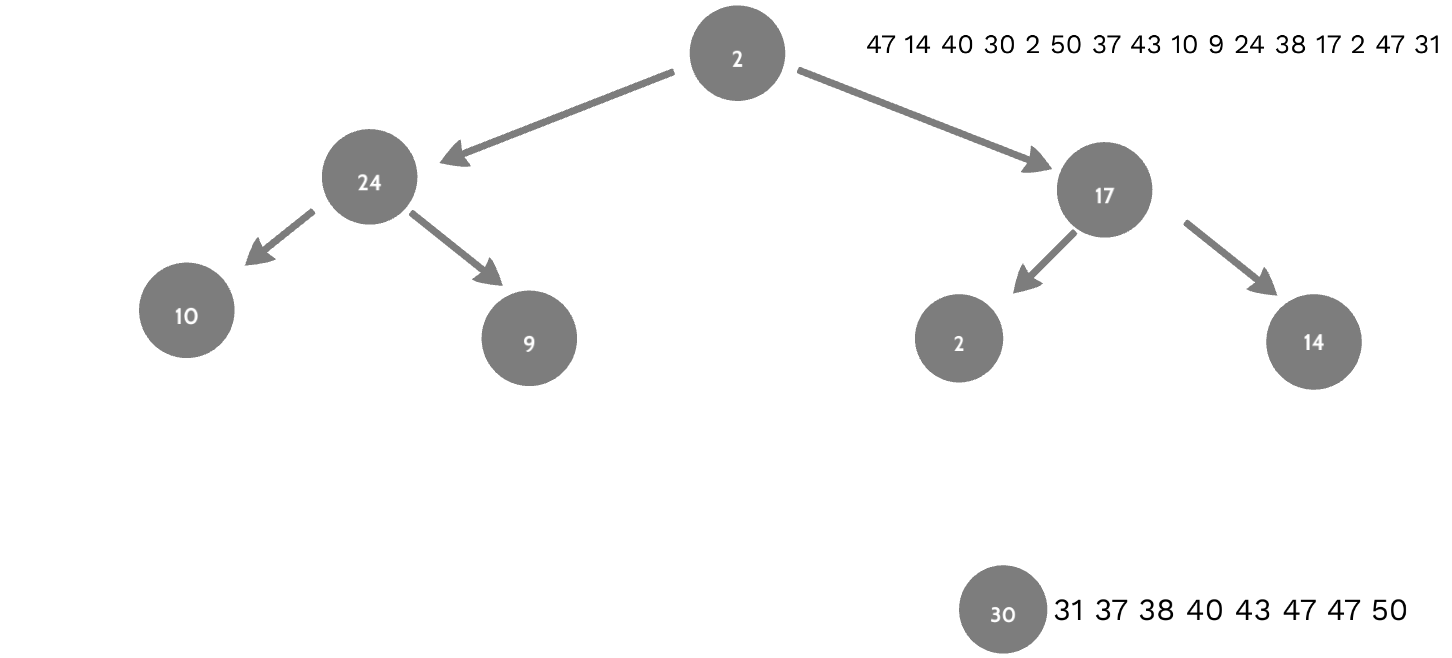
**----------------------------------------------------------------------------**

****

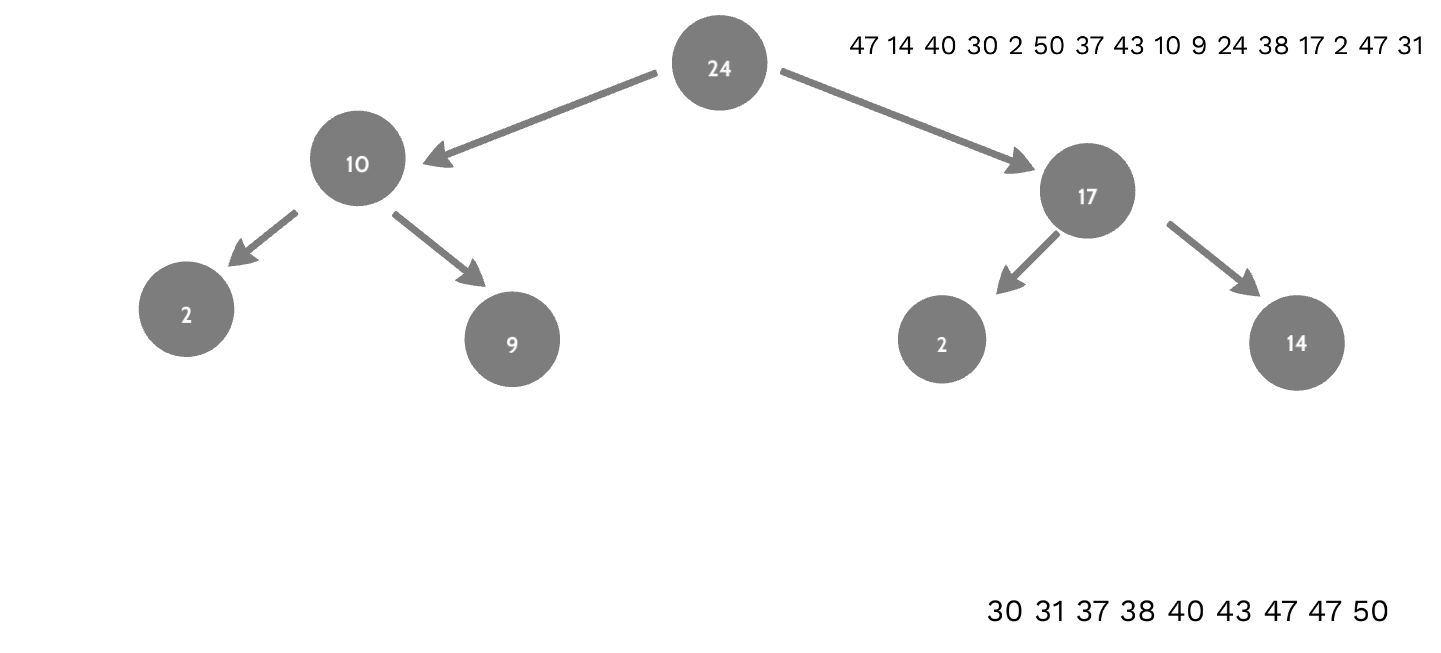
**----------------------------------------------------------------------------**

****

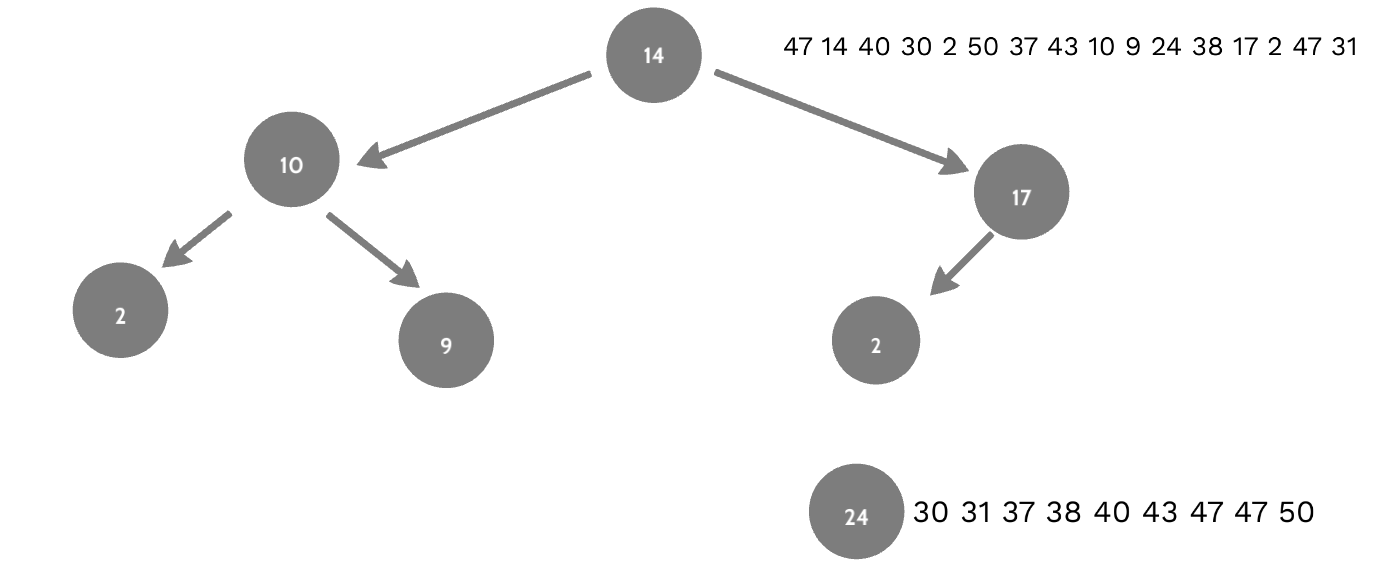
**----------------------------------------------------------------------------**

****

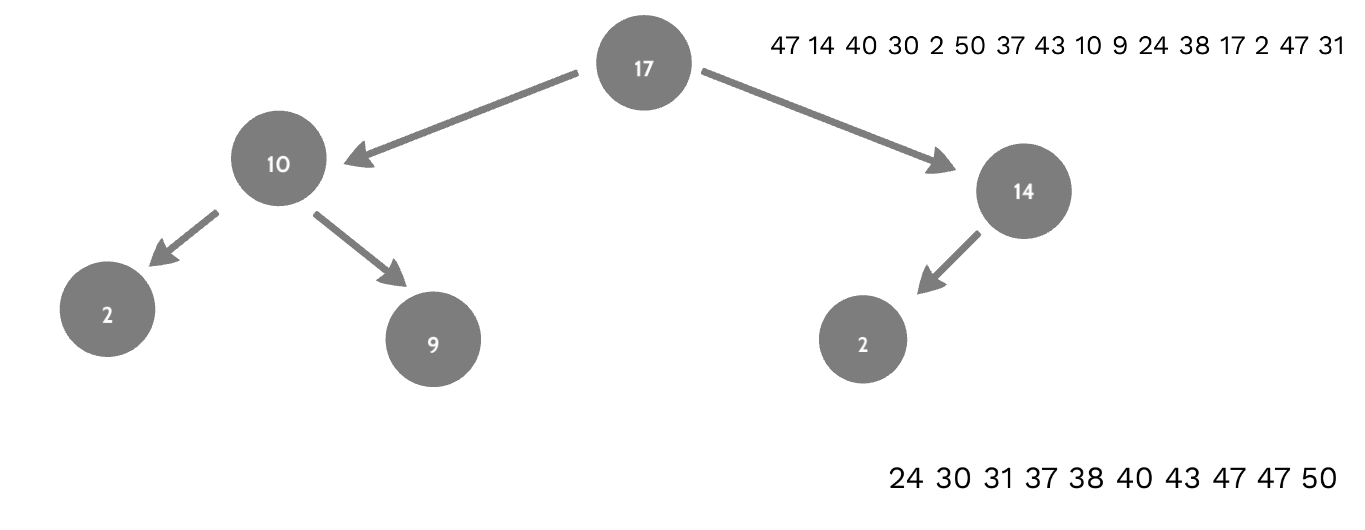
**----------------------------------------------------------------------------**

****

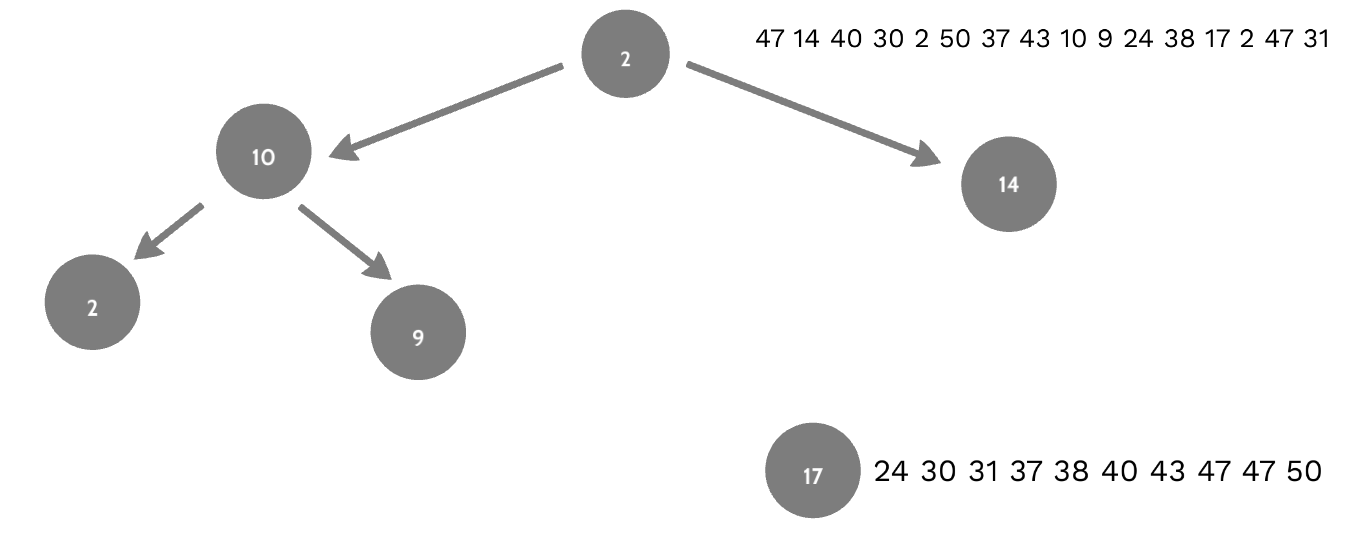
**----------------------------------------------------------------------------**

****

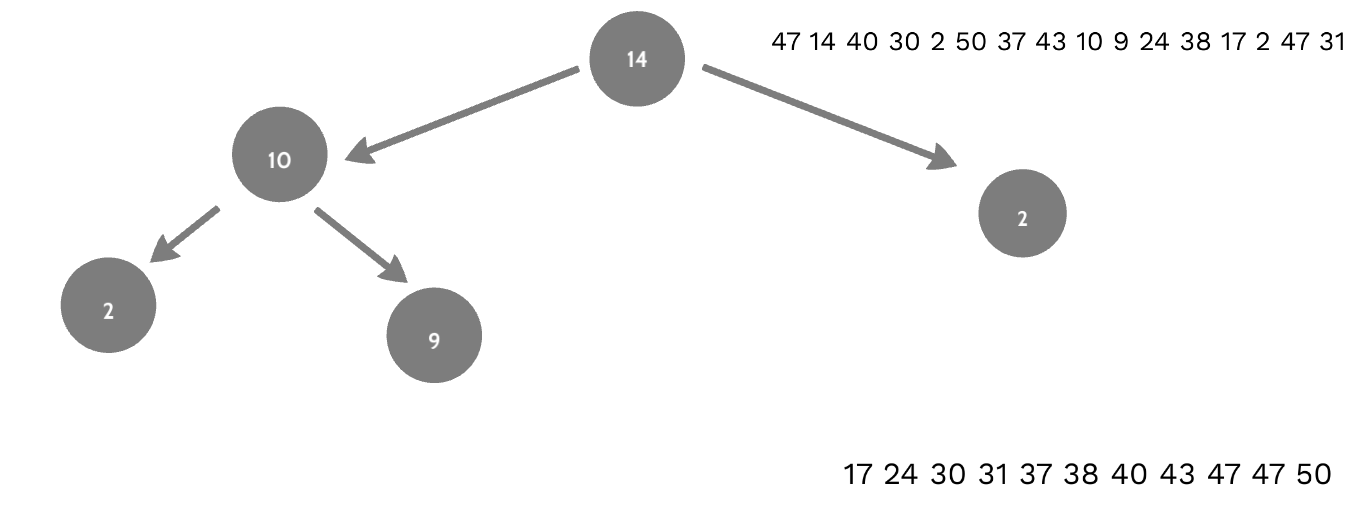
**----------------------------------------------------------------------------**

****

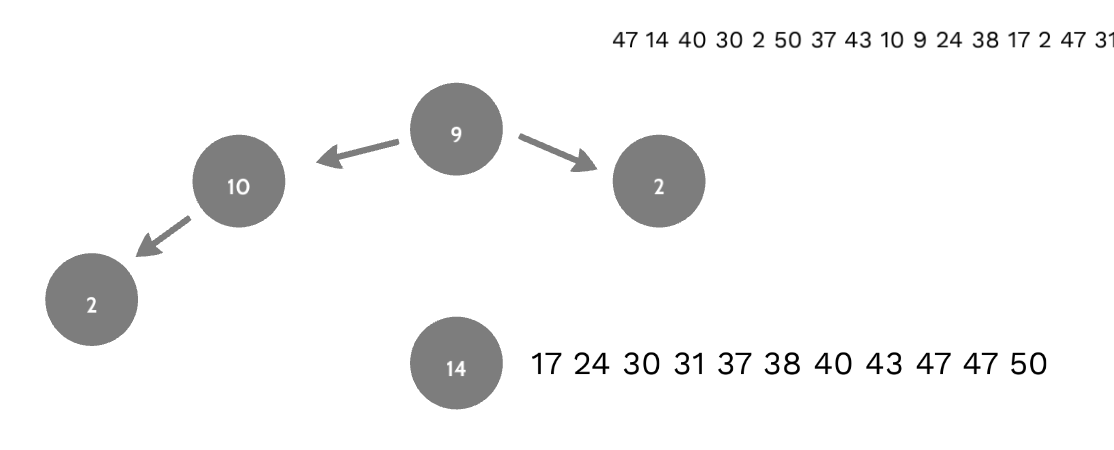
**----------------------------------------------------------------------------**

****

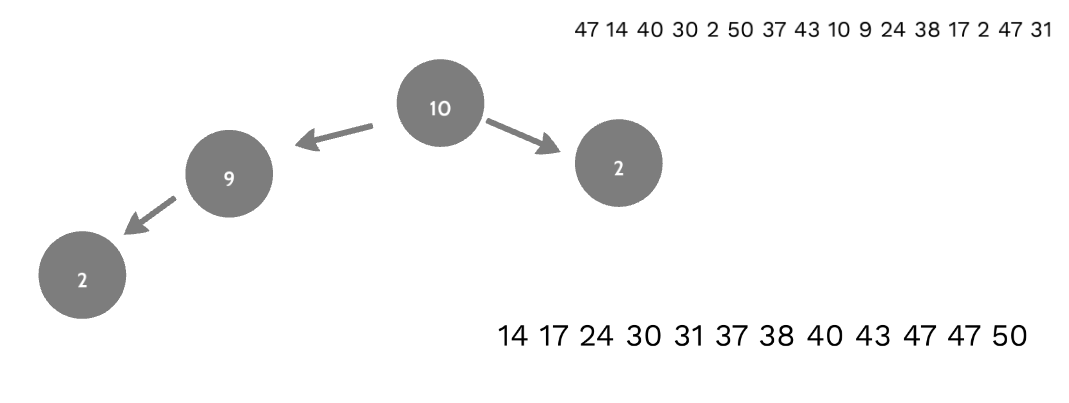
**----------------------------------------------------------------------------**

****

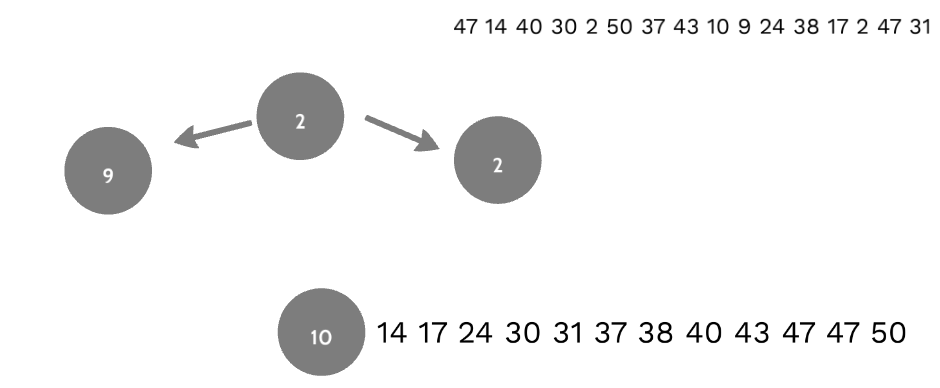
**----------------------------------------------------------------------------**

****

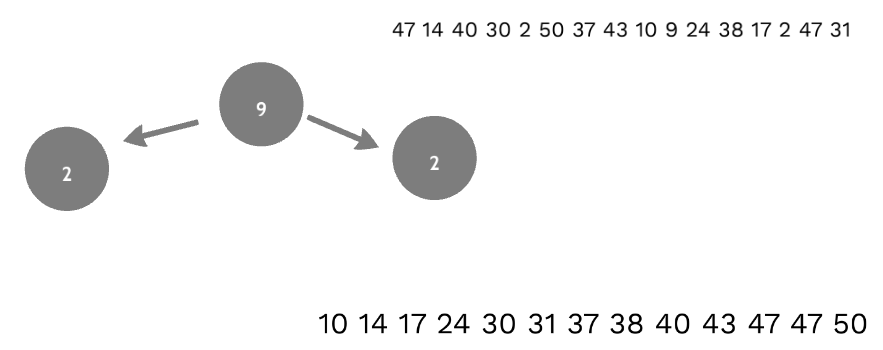
**----------------------------------------------------------------------------**

****

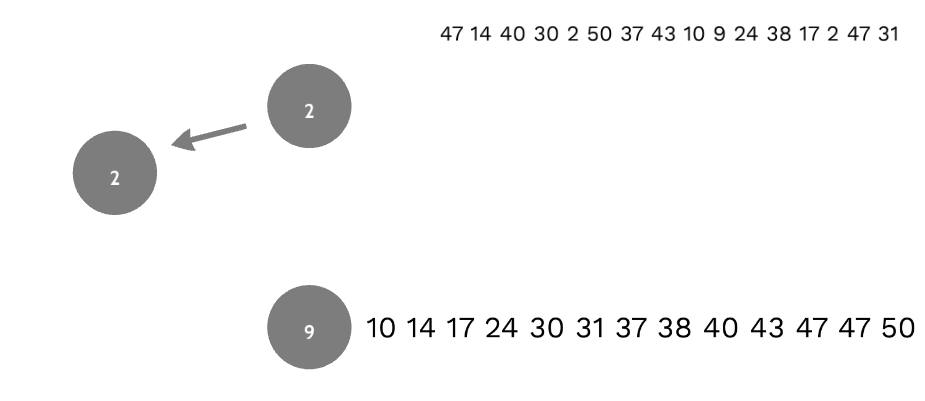
**----------------------------------------------------------------------------**

****

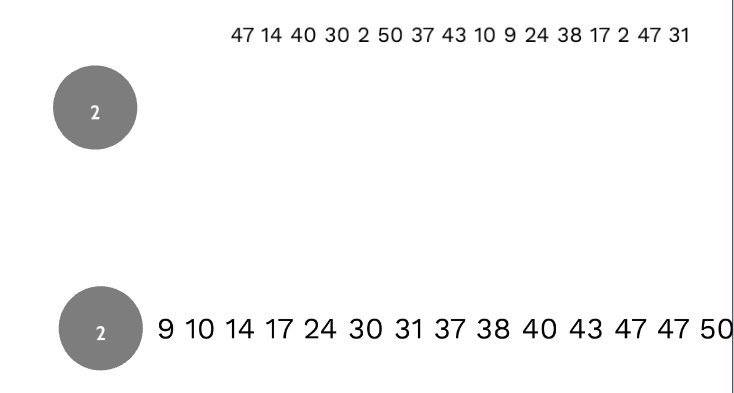
**----------------------------------------------------------------------------**

****

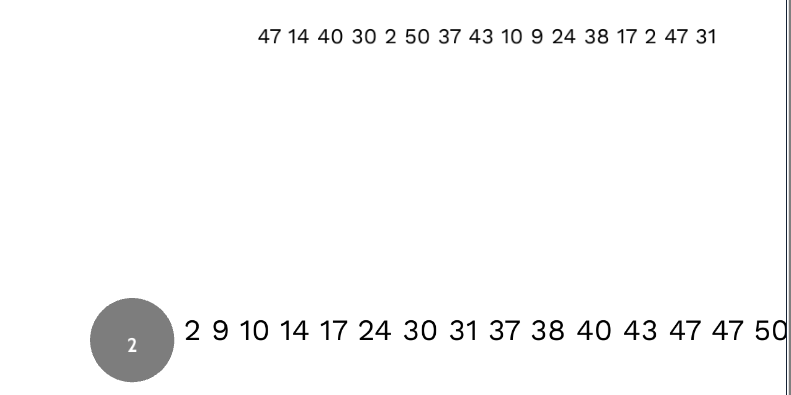
**----------------------------------------------------------------------------**

****

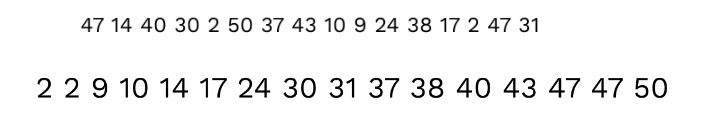
**----------------------------------------------------------------------------**

****

**----------------------------------------------------------------------------**

****

**----------------------------------------------------------------------------**

****

**----------------------------------------------------------------------------**

# Лістинінг пірамідального сортування (№4)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<math.h>

#include <conio.h>

#include <stdlib.h>

#include <ctime>

#include <time.h>

int main()

{

int a[1000], n, i, sh = 0, b = 0, t;

printf("n = ");

scanf\_s("%i", &n);

for (i = 0; i < n; ++i) {

a[i] = 0 + rand() % 101;

printf("%d ", a[i]);

}

printf("\n");

while (1)

{

b = 0;

for (i = 0; i < n; ++i)

{

if (i \* 2 + 2 + sh < n)

{

if (a[i + sh] > a[i \* 2 + 1 + sh] || a[i + sh] > a[i \* 2 + 2 + sh])

{

if (a[i \* 2 + 1 + sh] <= a[i \* 2 + 2 + sh])

{

t = a[i + sh];

a[i + sh] = a[i \* 2 + 1 + sh];

a[i \* 2 + 1 + sh] = t;

b = 1;

}

else if (a[i \* 2 + 2 + sh] < a[i \* 2 + 1 + sh])

{

int t = a[i + sh];

a[i + sh] = a[i \* 2 + 2 + sh];

a[i \* 2 + 2 + sh] = t;

b = 1;

}

}

}

else if (i \* 2 + 1 + sh < n)

{

if (a[i + sh] > a[i \* 2 + 1 + sh])

{

int t = a[i + sh];

a[i + sh] = a[i \* 2 + 1 + sh];

a[i \* 2 + 1 + sh] = t;

b = 1;

}

}

}

if (!b) sh++;

if (sh + 1 == n)

break;

}

for (i = 0; i < n; ++i)

printf("%i%c", a[i], (i != n - 1) ? ' ' : '\n');

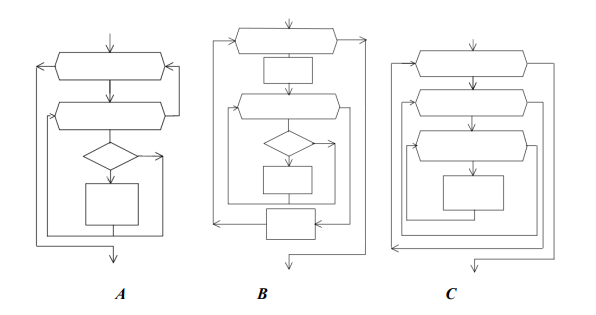
return 0;

}

****

# №5

5. Визначити яка схема належить якому методу сортування

****

А)Бульбашкове сортування

Б)Сортування вставками

С)Сортування вибором

***Висновки:*** я набув практичних навиків роботи з одновимірними масивами, а саме сортування елементів масиву різними методами. Здійснення порівняння та аналізу ефективності використовуваних методів сортування.