Reference: Big C++.

Exercises 11:

Exercise R11.1.

*int* min\_position(vector<*int*>& a, *int* from, *int* to)

{

*int* min\_pos = from;

*int* i;

for (i = from + 1; i <= to; i++)

if (a[i] < a[min\_pos]) min\_pos = i;

return min\_pos;

}

/\*\*

Sorts a vector using the selection sort algorithm

@param a the vector to sort

\*/

*void* selection\_sort(vector<*int*>& a)

{

*int* next; // The next position to be set to the minimum

for (next = 0; next < a.size() - 1; next++)

{

// Find the position of the minimum

*int* min\_pos = min\_position(a, next, a.size() - 1);

if (min\_pos != next)

swap(a[min\_pos], a[next]);

}

}

* For a vector of size = 0

1. Next = 0
2. a.size() = 0
3. a.size() – 1 = -1

The Vector is sorted, the loop will not be entered.

* For a vector of size = 1

1. Next = 0
2. a.size() = 1
3. a.size() – 1 = 0

The Vector is sorted, the loop will not be entered.

* For a vector of size = 2

1. Next = 0
2. a.size() = 2
3. a.size() – 1 = 1
4. The for loop is entered and min\_position is called with (a, 0, 1)
5. Inside the min\_position function, int min\_pos = 0, i = min\_position -> OK
6. to = 1 which is a.size() – 1, so in the for loop we need to check if i <= to
7. Vector is sorted.

Exercise R11.2.

Searching is about finding a value in a data structure. Sorting is about searching for the smallest value (is ascending sort) and sort the vector.

Exercise R11.3.

1. , For big values of n, 0.001 >>>

Exercise R11.4.

Exercise R11.5.

1000 record -> 5 sec

2000 record -> 10 sec

10000 record -> 50 sec

Exercise R11.6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 1000 | **5** | **5** | **5** | **5** | **5** |
| 2000 | **10** | **20** | **40** | **11** |  |
| 3000 | **15** | **45** | **135** | **17.38** |  |
| 10000 | **50** | **500** | **5000** | **66.67** |  |

Exercise R11.7.

Exercise R11.8.

Binary search is of order:

Exercise R11.9.

Exercise R11.10.

Exercise R11.11.

The cost of sorting the vector is , the cost of removing duplicates is So the total cost is which is not faster than

Exercise R11.12.

void remove\_duplicates(std::vector<int>& a)

{

int n = a.size();

std::vector<int> count(n, 0);

std::vector<int>::iterator i = a.begin();

while (i != a.end())

{

if (count[\*i % n] > 0)

i = a.erase(i);

else

{

count[\*i % n]++;

i++;

}

}

}