# C++ Programming: Judge Assignment 3 (JA3)

The following tasks should be submitted to the SoftUni Judge system, which will be open starting Saturday, 2 May 2017, 10:00 (in the morning) and will close on Sunday, 14 May 2017, 23:59. Submit your solutions here: <https://judge.softuni.bg/Contests/547/Judge-Assignment-3-JA3-Algorithms-STL-Data-Structures>

Solutions for each task will be submitted in the form of compressed archive (.zip) files, containing .h and .cpp files.

Please be mindful of the strict input and output requirements for each task, as well as any additional requirements on running time, used memory, etc., as the tasks are evaluated automatically and not following the requirements strictly may result in your program’s output being evaluated as incorrect, even if the program’s logic is mostly correct.

You can use C++03 and C++11 features in your code.

Unless explicitly stated, any integer input fits into int and any floating-point input can be stored in double.

NOTE: this assignment is focused on data structures and algorithms. Some tasks require trivial algorithms, but others require efficient usage of data structures and finding optimum algorithms to fit into the time and memory constraints. If you are having difficulties with thinking up an efficient algorithm or data structure – check the STL documentation on running time of various data structures, look up similar problems or sub-problems of the task you’re solving on the Internet, try to find standard algorithms which solve those sub-problems efficiently and combine them – you could also ask for hints or guidance in the forum (<https://softuni.bg/forum/categories/42/cplusplus-programming>), as long as you don’t expect someone to solve the tasks for you.

NOTE: the tasks here are NOT ordered by difficulty level.

## Task 3 – Minerals (JA3-Task-3-Minerals)

In the strategy game ZtarKrapht (*yep, shamelessly avoiding copyright here*), players need to collect "minerals" to build buildings and units, which they use to win the game. The game is played in a grid (i.e. a matrix) with coordinates from 0, 0 to 1000, 1000. Objects in the game (units, buildings, minerals) are placed in one of the coordinates of the grid.

To collect minerals, a player must place a building, called a "command center", at a certain position in the grid. Then a worker unit starts gathering minerals by moving from the position of the command center to the position of the closest mineral, taking it and going back to the command center. Then the worker repeats the process for the next closest mineral (the first one is removed from the grid after being collected) and so on.

The worker always picks the shortest possible path when moving from one position to another. The worker can move only up, down, left and right on the grid, with a speed of 1 grid cell per second. That means that if the command center is at position x, y and a mineral is at position p, q, then the time the worker needs to get from the command center to the mineral can be calculated by the formula t = abs(x - p) + abs(y - q) (look up Manhattan distance to see why). The time for going back is the same, so the total time to collect the mineral is 2 \* t.

Write a program, which, given the position of the command center, the positions of all the minerals in the grid, and a number M, calculates the minimum time for which M minerals can be collected (each time a mineral is collected it is removed from the grid).

### Input

The first line will contain the integer number M – the number of minerals to be collected.

The second line will contain two integer numbers, X and Y – the position of the command center.

The third line will contain the integer number N – the total number of minerals in the grid.

The next N lines will each contain two integer numbers – the position of a mineral in the grid.

### Output

A single line containing the minimum time for which M minerals can be collected.

### Restrictions

0 < M <= 5000; 1 < N <= 500000; M < N;

In 40% of the test cases M <= 30, N <= 200. In the other 60% of the test cases M \* 100 <= N;

The total running time of your program should be no more than 1.7s

The total memory allowed for use by your program is 16MB.

### Example I/O

|  |  |  |
| --- | --- | --- |
| Example Input | Expected Output | Explanation |
| 1  5 6  2  7 6  5 4 | 4 | We need 1 mineral. Both minerals are the same distance = 2 away, so whichever we pick we get a time of 4 |
| 3  0 0  5  4 4  2 0  0 1  2 2  3 3 | 14 | Closest 3 minerals are those at (0,1), (2,0), and (2,2). The distance to the first is 1, to the second is 2 and to the third is 4, so the time needed is (1+2+4) \* 2 = 14 |