1 Introduction

The presented algorithm strongly relies on the assumption, that we are not allowed to accommodate two workshops in one room (otherwise the problem would be in NPC). Firstly, we sort our rooms by the length of time, at which they are at our disposal. Secondly, we iterate through all of our rooms starting from the one, that is the shortest available room and ending at the longest available one, trying to accommodate in each of them a suitable (that has not yet been allocated and is fitting both with time and the number of participants) workshop with the largest number of participants possible. Meanwhile, whenever we accommodate a workshop in a room, we decrease the count of open-air workshops by one and the number of people, who will take part in them by the number of participants of this particular workshop.

Lemma

The algorithm described above finds an optimal (according to our problem) allocation of the workshops.

The proof of our Lemma

We look through our rooms in a way desribed above, keeping in mind, that they are sorted appropriately: If we encounter a room R, for which the cardinality of the set S of suitable workshops is greater than zero, we notice, that:

1. Having no workshop from S accommodated in room R we will not obtain a better solution to our problem, than having it done. Performing no accommodation simply decreases the number of available rooms, while the number of workshops remains the same. Hence we should allocate one of the workshops from S in the room R.

2. Accommodating such a workshop from S in room R, that has the greatest number of participants is optimal - all workshops from S fit with their duration time into next (in our iteration order) rooms after R, so the best thing to do is to locate the workshop of the greatest number of participants in the room R.

If the cardinality of S for considered room R is equal zero, we move to the next room and look for workshops suitable for it - no workshop from the ones, that haven't been allocated yet fits into room R. On the other hand, none of the workshops allocated before should change its room and be located in R, because at the moment it was accommodated, we did the best choice possible (according to points 1). and 2).).

Since each time we allocate a workshop in a room, we decrease appropriately the number of unallocated workshops (initially equal the number of all workshops) and the number of people, that will take part in open-air workshops (initially the number of all people participating in all workshops) - at the end of the algorithm we receive both values, that are asked for in our task.

2 Tests

The tests are stored in the following files:

- *test0* - example test
- *test1* - correctness test, one workshop fitting in one room
- *test2* - correctness test, one workshop too long to fit in one room
- *test3* - correctness test, four workshops, three of them fitting in three rooms
- *test4* - correctness test, nine workshops fitting in nine rooms
- *test5* - correctness test, two workshops, one of them fitting in one of two rooms
- *test6* - performance test, 10 trials, each containing 300 workshops and 300 rooms
- *test7* - performance test, 10 trials, each containing 1000 workshops and 300 rooms
- *test8* - performance test, 10 trials, each containing 300 workshops and 1000 rooms
- *test9* - performance test, 10 trials, each containing 1000 workshops and 1000 rooms
- *test10* - performance test, 10 trials, each containing 1000 workshops and 1000 rooms

Every correctness test has been hand-written, every performance test has been randomly generated by the (included) gentest.cpp program.