CS431 : Assignment - 3

Functional Programming with Haskell

Devaishi Tiwari (170101021)

3. Minimum Number of Moves

3.1 Write the algorithm (in pseudo-code) that you devised to solve the problem.

The algorithm consist of the following steps:-

1. Using the specified minimum and maximum dimensions, I found out all possible dimensions for each type of room.
2. Next, I paired the possible dimensions of each type of room to form a tuple.
3. I simultaneously kept on filtering out the tuples whose total area exceeds the maximum area, and the conditions placed on dimensions of kitchen and bathroom do not satisfy.
4. I also simultaneously remove the extra tuples which would lead to the same value of partial area to fasten the computation.
5. Once I get all the possible 6-tuples of dimensions, I find out maximum area and the dimensions corresponding to this area.
6. Finally, I print the result on the standard output (terminal).

3.2 How many functions did you use?

The algorithm consist of the following major functions:-

1. **getDimensions & getDimensionsWidth** - Using the specified minimum and maximum dimensions, finds out all possible dimensions for each type of room.
2. **make2Tuple, make3Tuple, ... , make6Tuple** - pairs the possible dimensions of each type of room to form a tuple.
3. **filterDimensions1, ... , filterDimensions1** - removes the extra tuples which would lead to the same value of partial area to fasten the computation.
4. **findMaximum** - finds out maximum area possible with the given constraints.
5. **Design** - The main function, filters out the tuples whose total area exceeds the maximum area, and whose conditions placed on dimensions of kitchen and bathroom do not satisfy. It also prints the result on the standard output (terminal).

3.3 Are all those pure?

Yes, all the functions used in this problem are pure. This is because there were no I/O and randomization processes that could change the state of the function

4. Common Questions

4.1 Do you think the lazy evaluation feature of Haskell can be exploited for better performance in the solutions to the assignments? If so, which solution(s) and how?

The lazy functionality to the Haskell programming language has a number of benefits. It is most useful in computation over large sized lists (infinite lists even).

In the third question, there were a large number of possible cases. Hence, I used lazy functionality of Haskell to fasten up the process creatively.

4.2 We can solve the problems using any imperative language as well. Do you find any advantage of using Haskell for these problems (w.r.t the property of lack of side effect)? If your answer is no, elaborate on why not?

The two major features of Haskell that gives it an edge over other languages are the **lazy functionality** and the **no side-effect functionality.**

Lazy functionality helps in avoiding unnecessary calculations, as the values are computed as and when they are needed. No side-effect functionality means that the values of variables are unchangeable. This has the following benefits,

- Easy to debug the code.
- Easy parallelism as no critical sections present