Midterm 2 – minimum stack

Table of Contents

Files provided 1
Problem Statement 1
Example 1
Algorithm using two stacks 2
Testing 4
Submitting your solution 4
Files to submit 4
Due and Grade 4

Files provided

- minstack.pdf this file
- drivermin.cpp a skeleton code
- driverminx.exe a solution for checking

Problem Statement

As we all know that most stack operations are working in constant time O(1) except some such as min() and max() in O(n) usually.

Here we want to Implement a helper stack that supports push(), pop(), top() as usual, and an additional min() operation which returns **the minimum element from the helper stack.** All these operations of the helper stack must be in constant time, O(1). To implement the helper stack, use the liststack.cpp provided and no other data structure like arrays, vectors, etc.

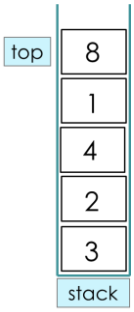
- Do not modify the existing `liststack.cpp, driver.cpp` or `liststack.h` files, but augment the existing `driver.cpp` code such that it supports the algorithm. Use `drivermin.cpp` file provided that is basically a copy of `driver.cpp.`

- You may do this work in **pset8a** folder where you have worked on `liststack.cpp` since this part needs to build with the file.

Example

Let us suppose we have the following elements are pushed in a stack.

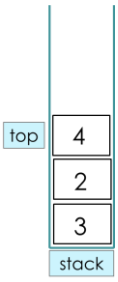
push(3)
push(2)
push(4)
push(1)
push(8)



When min() is called for the helper stack, it should return 1, which is the minimum element in the current stack.

Then, we pop twice on the stack, the stack becomes

pop()
pop()



When min() for the helper stack, it should return 2 which is the minimum in the current stack.

Algorithm using two stacks

Use two stacks: one to store actual stack elements and the other as an auxiliary helper stack to store minimum values, based on the actual stack.

The idea is to do push() and pop() operations in such a way that the top of the auxiliary helper stack is always the minimum. Let us see how push() and pop() operations work. Let us call the actual stack as '`stack`' and the auxiliary helper stack as '`minstack`'.

- push(int x)
  - push x to the `stack` (the stack with actual elements)
  - compare x with the top element of the `minstack`. Let the top element be y.
    - the stack is not empty and If x is greater than y then push y to the `minstack.`
    - otherwise push x to the `minstack` (since x is the minimum)
- pop()
  - pop the top element from the `minstack`. (discard the element.)
  - pop the top element from the stack and return it.

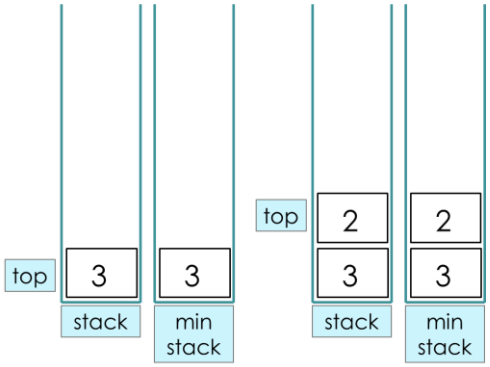
- min()
  - min() returns the minimum element from the `minstack.`

The min() function is called in m option in our code. Besides these operations, you should write necessary coding for **o, P, O and c** options etc.. You must figure them out since you may **not** find the kind messages such as "your code here".

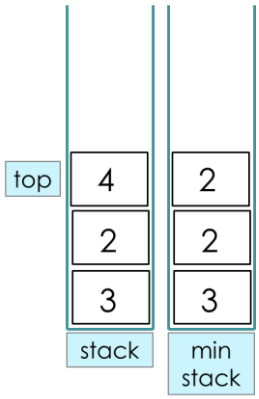
We can see that all the above operations are O(1).

Let us see an example. Let us assume that both stacks are initially empty and 3, 2, 4, 1, and 8 are inserted to the `minstack`

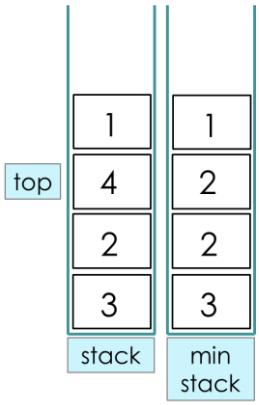
When we insert 3 and 2 in order, both stacks change to the following:



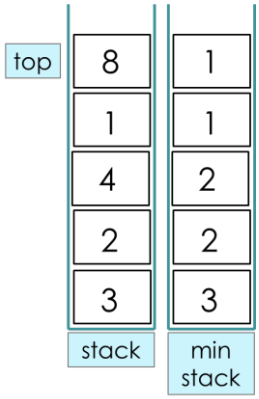
When we insert 4, both stacks change to the following since 2 is the minimum:



When we insert 1, both stacks change to the following since 1 is the minimum:



When we insert 8, both stacks change to the following since 1 is the minimum:



Testing

You may use p and P options to compare the actual timing using with P options multiple times. Try the number of nodes and over several hundred millions to see the difference. Elapsed time for O(1) should prints 0 sec on the console.

Submitting your solution

- **On my honour, I pledge that I have neither received nor provided improper assistance in the completion of this assignment.**
  Signed: _____ Section: _____ Student Number: _____
- Make sure your code **compiles** and **runs** right before you submit it. Don't make "a tiny last-minute change" and assume your code still compiles. You will not receive sympathy for code that "almost" works.
- If you only manage to work out the Project problem partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- Place your source files in the folder you and I are sharing.
- After submitting, if you realize one of your programs is flawed, you may fix it and submit again as long as it is **before the deadline**. You will have to resubmit any related files together, even if you only change one. You may submit as often as you like. **Only the last version** you submit before the deadline will be graded.

Files to submit

Submit the following files on Piazza folder.

- **drivermin.cpp**

Due and Grade

- Due: 11:55 pm