National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms

Midterm Test @ LT15+LT19

(Fri, 23 Feb 2018, S2 AY2017/18, 60m)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains FIVE (5) sections.
   It comprises EIGHT (8) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.
7. Write your Student Number in the box below:

A	0	1						

Section	Maximum Marks	Average ± Stdev	Remarks
A	10		
B	15		
C	30		
D	40		
E	5		
Total	100		

A Worst Case Time Complexity Analysis (10 marks)

Write down the *tightest*[1] *worst case* time complexity of the various data structure operations or algorithms below.

The operations (algorithms) referred below are the **unmodified version**, as per discussion in class, e.g. as currently explained in VisuAlgo or as currently implemented in C++ STL. Unless otherwise mentioned, there are currently n elements in the data structure.

No	Operations		Time Complexities
1	Insert a new element at the *second last* position of a `std::vector`		$O($_____$)$
2	Erase the *first* element of a **sorted** `std::vector`		$O($_____$)$
3	Compare if two **equal-size** `std::vector`s have the same elements		$O($_____$)$
4	Run `std::stable_sort` on a `std::vector`		$O($_____$)$
4	Search for the **first occurrence** of value `v` in a **sorted** `std::vector`		$O($_____$)$
6	Reverse the current content of `std::list`		$O($_____$)$
7	Get the *third last* element of `std::list`		$O($_____$)$
8	Get the *second from top* element of `std::stack`		$O($_____$)$
9	Get the *second last* enqueued element from a `std::queue`		$O($_____$)$
10	Get the *i-th element* of an `std::deque`		$O($_____$)$

[1] What we meant by tightest worst case time complexity is as follows: If an operation of the stipulated data structure/an algorithm needs at best $O(n^3)$ if given the worst possible input but you answer higher time complexities than that, e.g. $O(n^4)$ – which technically also upperbounds $O(n^3)$, you will get wrong answer for this question.

Section A Marks = _____

B Analysis (15 marks)

Prove (the statement is correct) or disprove (the statement is wrong) the statements below.

1. We always have to use C++ `cin/cout` for I/O and C++ `class` instead of using C `scanf/printf` and C `struct`, otherwise `g++` compiler (assume gcc 5.4.0 that is currently used by Mooshak Online Judge) will raise a compilation error.

2. There is a possibility that the implementation of C++ STL `std::stable_sort` uses Randomized Quick Sort, one of the fastest known sorting algorithms in the world.

3. A Doubly Linked List (DLL) uses more memory per vertex than a Singly Linked List (SLL).

4. If we implement Stack ADT with C++ STL `std::vector` as the underlying data structure (with its's back as the top side of the stack), we can get the value of *any* element (top, second from top, bottom :O, etc) currently inside the stack in $O(1)$.

5. As C++ STL `std::deque` is a superset of `std::vector`, we can *generally* replace `std::vector` inside a C++ code with `std::deque` and it will work just fine (i.e. it compiles).

Section B Marks = _____

C Alternative Implementations (30 marks)

C.1 Non-Compact std::vector Implementation for List ADT (15 marks)

In class, we have discussed possible implementations of List ADT using either compact array (fixed-size) or compact `std::vector` (variable-size). For this question, assume that we use a variable-able `std::vector`. Someone suggested that we can potentially speed-up the `remove(i)` operation for List ADT into $O(1)$ for all cases of `i` $\in$ `[0..N-1]` if we do **not close the resulting gap** after removal of the `i`-th element of the list. This makes the underlying `std::vector` non-compact. Now, **what are the implications of such implementation**? One such implication has been listed below. Your job is to list down *at least 5 (can be more...)* other logical statements of this implementation. Your answer will be graded based on the soundness and the quality of the statements (3 marks for each valid statement; -1 mark for random/irrelevant statement; min 0 mark and max 15 marks).

1. To implement `get(i)`, we now have to count the number of non-blank elements starting from index 0. This is $O(N)$ instead of $O(1)$.
2.
3.
4.
5.
6.
7. (optional):

C.2 Singly/e Linked List without Tail Pointer (15 marks)

In class, we have an implementation of Singly/e Linked List (SLL) with head **and tail** pointers. Someone said that it is not useful to also store (and maintain) tail pointer to always point to the last element of the list. Now, **what are the implications of such implementation**? One such implication has been listed below. Your job is to list down *at least 5 (can be more...)* other logical statements of this implementation. Your answer will be graded based on the soundness and the quality of the statements (3 marks for each valid statement; -1 mark for random/irrelevant statement; min 0 mark and max 15 marks).

1. The most obvious implication is that to get the tail element, we have no choice but to start from the only vertex that we know (the head element) and iterates forward N-1 times to reach the tail element. This is $O(N)$ instead of $O(1)$.
2.
3.
4.
5.
6.
7. (optional):

.

Section C Marks = _____ + _____ = _____

D Applications (40 marks)

D.1 Special Sorting Criteria (20 marks)

You are given a list of names (each name consists only of lowercase alphabet ['a'..'z'] and is not more than $M = 100$ characters) ending with an End of File (EOF) marker. There are N names/lines in the input and $1 \leq N \leq 10\,000$. Your job is to sort the names based on how they ends (suffix) rather than how they start (prefix). If two words have the same suffix, the shorter word is sorted earlier. Output these N sorted names in N lines.

Sample Input	→	Sample Output
ranaldlamyunshao		leminhphuc
melvintanjunkeong		ong
leminhphuc		melvintanjunkeong
bookaihsien		hoangduong
tanjunan	→	sidhantbansal
hoangduong		tanjunan
sidhantbansal		bookaihsien
ong		ranaldlamyunshao

A skeleton C++ code has been written for you. Please complete it and analyze its time complexity.

```
#include <bits/stdc++.h> // you have to complete this question using C/C++ code
using namespace std;

int main() {
  string name;

  while (cin >> name, !cin.eof()) { // if can read name in one line (not EOF yet)




  }









  return 0;
} // the overall time complexity of my C++ code above is O(_____________)
```

D.2 PS1 - Continuous Median (version IV) (20 marks)

You are given exactly the same problem (Continuous Median) as with PS1 with the following changes: $TC = 1$ (that is, only one test case), $1 \leq N \leq 20\,000\,000$ (2×10^7), and $1 \leq A[i] \leq 3\ \forall i \in [0..N\text{-}1]$.

Sample Input	→	Sample Output
1 6 1 2 3 1 2 3	→	9

A skeleton C++ code has been written for you. Please complete it and analyze its time complexity. Note that in PS1 - version III with $1 \leq TC \leq 3; 1 \leq N \leq 100\,000$, one needs an $O(N \log N)$ solution to pass the 1s time limit. PS: The full statement of PS1 version I is displayed on LT15+LT19 projectors.

```
#include <bits/stdc++.h> // you have to complete this question using C++ code
using namespace std;
int main() {
  ios::sync_with_stdio(false); cin.tie(NULL); // N is gigantic, we need fast I/O
  int TC, N, Ai;
  cin >> TC; // always 1 test case for this version IV
  cin >> N;
  long long ans = 0;



  for (int i = 0; i < N; i++) {
    cin >> Ai; // this time, 1 <= Ai <= 3









  }
  cout << ans << endl;
  return 0;
} // the overall time complexity of my C++ code above is O(_____________)
```


Section D Marks = _____ + _____ = _____

E Easy Marks (5 marks)

Write a **short** (maybe limit yourself to just 2-3 minutes to do this) **but honest** feedback on what you have experienced in the first 6 weeks of CS2040C. Suggestions that are shared by majority (not a one-off feedback) and can be easily incorporated to make the next 6 weeks of CS2040C better (Week 07-12) will be done. Grading scheme: 0-blank, 1/2-considered trivial feedback but not blank, 4/5-good and constructive feedback, thanks.

– End of this Paper, All the Best –