All problems are written problems; submit your solutions electronically as a PDF files via UMlearn. There are 92 marks available. The assignment will be marked out of 90. Please read http://www.cs.umanitoba.ca/~kamalis/comp3170/info.pdf for guidelines on academic integrity.

Problem 1 Special Stack Variant [10 marks]

In the class, we saw a special type of stacks in which each operation involves popping \( n \geq 0 \) items followed by pushing exactly one item. Consider a variant in which each operation involves popping one item followed by pushing \( n \geq 0 \) items. Assume the stack is implemented using an array of fixed size \( C \) and the number of items in the stack is never more than \( C \).

Use the potential function method to show the amortized cost of each operation is at most 2.

Answer: Define the potential to be the number of empty cells in the array. For an operation involving \( n \) pushes, the actual cost is \( n + 1 \) and for each push, the potential is decreased by 1. So, the difference in potential will be \( -n \) for pushed items and +1 for the pop, i.e., a total difference of \( -n + 1 \). The amortized cost will be \( (n + 1) + (-n + 1) = 2 \).

Scheme: 5 marks for the right potential function and 5 marks for finding the right amortized cost. You get the full mark as long as you show the amortized cost is constant via a potential function argument.

Problem 2 Kruskal’s MST Algorithm [15 marks]

In the class, we saw that disjoint sets are used to keep track of the connected components in the Kruskal’s MST algorithm. For the graph of Figure[1] trace the algorithm by showing the connected components after processing each edge. You do not to draw any graph (just indicate the sets). Initially, the components are \( \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\} \), which form Line 0 of your answer. In each of the following lines, indicate the sets associated associated with connected components after processing the subsequent edge. So, the \( i \)’th line of your solution might looks like:

\[
\text{processing edge } (x,y) \rightarrow \{u,v\}, \{w,x,y\}, \{z\}
\]

Answer: \( \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\} \)

\[
\text{processing edge } (a,d) \rightarrow \{a,d\}, \{b\}, \{c\}, \{e\}, \{f\}, \{g\}
\]
processing edge (b,g) → \{a, d\}, \{b, g\}, \{c\}, \{e\}, \{f\}
processing edge (e,f) → \{a, d\}, \{b, g\}, \{c\}, \{e, f\}
processing edge (a,b) → \{a, d, b, g\}, \{c\}, \{e, f\}
processing edge (c,d) → \{a, d, b, g, c\}, \{e, f\}
processing edge (b,c) → [no change] \{a, d, b, g, c\}, \{e, f\}
processing edge (c,g) → [no change] \{a, d, b, g, c\}, \{e, f\}
processing edge (d,e) → \{a, d, b, g, c, e, f\}

Scheme: You lose 3 marks for each mistake at any given step.

**Problem 3 Union by Weight Analysis [10 marks]**

In this problem, we would like to show the amortized time of a union operation when union-by-weight on linked-lists is used is \(\Omega(\log n)\). For that, we need to come up with a sequence of \(\Theta(n)\) operations for which the amortized cost per operation is \(\Omega(\log n)\). We start with \(\text{make-set}(x_i)\) for \(i \in \{1, 2, \ldots, n\}\) where \(n\) is a power of 2. Provide a consequent sequence of \(\Theta(n)\) union operations so that the total number of updated pointers for all operations is \(\Omega(n \log n)\).

Answer: Here are the sequence of operations and their respective number of pointer-updates:

- **Step 1:** \(\text{union}(x_i, x_{i+n/2})\) for \(i \in \{1, \ldots, n/2\}\). There will be 1 pointer-update per operation, which sums to \(n/2\) total updates in this step.

- **Step 2:** \(\text{union}(x_i, x_{i+n/4})\) for \(i \in \{1, \ldots, n/4\}\). There will be 2 pointer-update per operation, which sums to \(n/4 \times 2 = n/2\) total updates in this step.

- **Step 3:** \(\text{union}(x_i, x_{i+n/8})\) for \(i \in \{1, \ldots, n/8\}\). There will be 4 pointer-update per operation, which sums to \(n/8 \times 4 = n/4\) total updates in this step.

- ...
• Step $k$: union($x_i, x_{i+n/2^k}$) for $i \in \{1, \ldots, n/2^k\}$. There will be $2^{k-1}$ pointer-update per operation, which sums to $n/2^k \times 2^{k-1} = n/2$ total updates in this step.

After $k = \lceil \log n \rceil$ steps, there will be most two sets, which are united with union($x_1, x_2$), with $n/2$ pointer updates. In summary, there will be roughly $\log n$ steps, each involving update of $n/2$ pointers. The total number of pointer-updates will be $\Omega(n \log n)$.

Scheme: 7 marks for showing a sequence of operations with $\log n$ steps each involving $\Theta(n)$ pointer-updates. 3 marks for the right conclusion about the amortized cost.

**Problem 4 Union-Find Operations [10+10 marks]**

• Consider a union-find structure based on union-by-rank and path-compression which is formed by $T_1$ and $T_2$ in the following figure. Draw the result after the following operations: union($T_1, T_2$), find(h).

![Diagram of T1 and T2](image1.png)

• Consider a similar structure formed by $T_3$ and $T_4$ in the following figure. Draw the result after the following operations: union($T_3, T_4$), find(k). For the union operation, as both trees have the same rank, assume $x$ becomes the parent of the united tree.

![Diagram of T3 and T4](image2.png)

Answer: See the following figure: