Preyasi Gaur Disc 1A

Time: 8:00AM - 9:50 AM

TA: Vincent Li

Computer Science 180

Homework 1

**Question 1**

1. Stability is not possible in this algorithm.
2. Proof by counterexample:
   * Suppose network A has the schedule with ratings: , and network B has the schedule:
   * Now notice in the current schedule, A wins spots 1 and 3, while B wins spot 2.
   * Now suppose network B’s schedule is changed to while Network A’s schedule remains the same. In this case B wins spots 1, and 2 while Network A wins spot 3.
   * Now suppose, network A’s schedule is changed to . This way, A wins spots 3 and 2, while B wins spot 1
   * Thus, we see that there isn’t any stable schedule as changing one of the schedules will win a greater number of spots.

**Question 2**

* Algorithm:
  + While hospital has open positions
  + offers a position to the following student, *s*, on it’s student ranking list
  + Now if is free (not been allotted to any other hospital), then accepts the offer
  + Else, if is committed to another hospital, ,
    - If prefers more then, it sticks with and rejects
    - Else if prefers more than , then it switches its choice from to , leaving an opening in , and decreasing the number of open positions in by one
* To Prove: the above algorithm is stable

We are given two cases of instability in the question and we will show that both of them cannot occur by proving by contradiction.

* + To prove this cannot happen: There are students s and s’, and a hospital h, so that s is assigned to h, and s’ is assigned to no hospital, and h prefers s’ to s.
    - Suppose there are two students 𝑠 and 𝑠’, and we have a hospital h.
    - Now according to the algorithm, if h prefers 𝑠’ to 𝑠, then h would have first offered to s’ before s.
    - Now, the only reason why s’ would reject hospital h, would be if it is paired with a hospital, h’ that it prefers more. Thus, the given instability cannot occur.
  + To prove this cannot happen: There are students s and s’, and hospitals h and h’, so that s is assigned to h, and s’ is assigned to h’, and h prefers s’ to s, and s’ prefers h to h’
    - Suppose h prefers s’ to s which means that it made an offer to s’ first.
    - Now the only reason s’ rejected the offer of h, was because it was matched with a hospital h’’ that it prefers more i.e. for s’ → h’’ > h.
    - As the algorithm continues, we see that the h’ is matched with s’ which means for s’ → h’ > h’’. Thus the rankings for s’ look like h’ > h’’ > h. This is contradictory as h matched to s’ causes the instability, which can only be possible if h > h’.

**Question 3**

Notice that this question, while framed differently, is very similar to the stable matching algorithms that we have encountered earlier.

* We need to find a stopping port, where the ship will stay for the rest of the month, for each ship’s schedule. These stopping ports are the defining factors of the truncations of the schedules. To satisfy the given condition we will set up a stable matching problem with the following constraints.
* Each ship has its own preference list (i.e. in which order it will visit ports) and as do the ports. The difference between the lists is that the ships rank each port in the order it visits them, and the ports rank in the reverse chronological order.
* Using these assumptions, we just need to create stable matching for them such that no two ships are on a port on the same day.
* Proof by Contradiction:
  + Assume that the assignment is not acceptable → there are at least two ships on the same port on the same day.
  + After running our stable matching algorithm let it be so that (s1, p1) and (s2, p2) are paired.
  + If a ship s2 passes through port p1 after ship s1 has already stopped there, which implies that in the preference list of s2 : p1 > p2
  + And if s2 passes through port p1 after ship s1 has already stopped there, then this pimples that in the preference list of p1, s1 > s2
  + In our case, now the unstable matching is (s2, p1). This means that ship s1 passes through the port p1 after ship s2 has already stopped there. But then in this case, under our preference relation, ship s1 prefers p1 to its actual stopping port and port p1, also prefers ship s1 to s2. But this is a contradiction to the fact that we chose stable matching pairs.
  + Moreover, we can also say that as we have transformed this problem into a classic stable matching problem, our Gale-shapley algorithm will always give us stable matching.

**Question 4**

We find the ordering to be:

Considering all the function and discussing their time complexities, we see:

* has a relatively slower growth rate because it has a square root of a logarithm which already grows slowly. Thus, as compared to all of the other given functions, this has the slowest growth rate.
* has a polynomial growth rate with a cubic term, which is slower than exponential growth but still much faster than
* ​​ has a polynomial growth rate with a fractional exponent. Moreover, considering and : 𝑙𝑜𝑔(𝑛) > 4/3 for the larger values of n
* has an exponential growth rate but as the exponent is log(n), it still does not grow as fast as linear, polynomial, or constant exponents and thus it’s growth rate is faster than the previous function but slower than the subsequent exponential ones.
* has simple exponential growth and increases rapidly with increasing n.
* has one of the fastest growth as it has an quadratic function as the exponent which makes it grow incredibly rapidly as n increases
* has the fastest growth rates as there is an exponential function as the exponent.

**Question 5**

1. To Prove: Prove that sum of the first n integers (1+2+....+n) is n(n+1)/2.

Proof: We will prove this using mathematical induction.

* Base Step:   
  Considering the base case, n = 1 we see that 1 = (1)(1+1)/2 = 1(2)/2 = 1.
* Inductive Step:  
  Now suppose that for an integer k, the sum of first k integers is   
  (1+2+...k) = k(k+1)/2 ……………….. (1)
* Then we add k+1 to both sides in (1) to get,

Thus, we conclude the inductive step and have proved that the sum of the first n integers (1+2+....+n) is n(n+1)/2.

1. Hypothesis to prove:

Proof: We will prove this using mathematical induction.

* Base Step:
* Considering the base case, n = 1, we see that . Thus the equation holds for the base case.
* Inductive Step:   
  Now suppose that for a natural number k, the sum of the squares of the first k natural numbers is, ……………….. (1)
* Then we can add on both sides of (1) to get,

Thus, we conclude the inductive step and have proved

**Question 6**

* Step 1: The above problem could be solved by storing the frequencies of all of the elements in the array in a hashmap data structure. For the hashmap, the key will be the value of the number, and the val would be the frequency of that number.
* Step 2: We store the frequencies of each number in another hashmap , and keep track of the minimum frequency (min\_freq) among all the frequencies. This process would take O(n) time as we have to check n frequencies to track the minimum.
* Step 3: We define a variable min\_num to track the minimum number with the minimum frequency. Then we linearly traverse the second hashmap (with all the frequencies) to see which key with the frequency of min\_freq is smallest by comparing min\_num and the value of the key. This take O(n) time If we get a smaller value then we can set min\_num = value of key
* After step 3, we return min\_num
* Time complexity: O(n) (Step2) + O(n) (Step3) = O(n)