CSE548 Fall 2018 Analysis of Algorithms

Ayush Garg (SBU ID 111870086)

September 16, 2018

1 Question 1

1. (a) We are given with k=2 jars and n floors. With the given information, we have to determine a strategy where we can drop a jar at most f(n) times where f(n) should grow slower than linearly(worst case no. of drops). i.e. $\lim_{n\to\infty}$ f(n)/n=0.

   Strategy: The basic idea is to divide n floors into some optimally sized m blocks such that we can first go and try dropping from top of each of the blocks starting from block 1 to block m. Once the glass breaks from a block, we know that this is the block in which our answer must lie. Then we go linearly upwards the number of floors in that block. When our second glass break, we know that the floor below that is our answer. Now the real question is to choose the value of m. We observe that if the value of n is chosen as 2. Then we drop the glass from (n/2)th floor. If the glass breaks, then we start linearly from 1 to n/2. If the glass does not break then we start linearly from (n/2)+1 to n. But f(n) will be O(n/2) ≡ O(n) in this case. Similarly if we choose m to be 10, f(n) will be O(n/10)≡O(n). Now suppose we choose m to be $\sqrt{n}$ then n would be divided into$\sqrt{n}$, $\sqrt{n}$ sized blocks and $\sqrt{n}$ linear steps, which looks ideal. In this case, We first drop a glass from 1st block. If it doesn't break, we go all the way from 1st to $\sqrt{n}$th block. The moment it breaks, we know that this is the block we are looking for and we use our second jar to linearly drop inside that block. In worst case, the number of drops would be $\sqrt{n}$+$\sqrt{n}$=2$\sqrt{n}$ which is O($\sqrt{n}$). Also, we can show that $\lim_{n\to\infty}$ $\sqrt{n}$/n=0.

2. (b)Using the concept of a), we can devise a similar strategy when k¿2 jars. Obviously our number of drops should be lesser when k increases. The best case is when we can use binary search O($\log n$). Now if we have k¿2 blocks, then we can further divide the $n^{\frac{1}{2}}$ sized block where our glass broke, before proceeding with the linear search. Hence we will observe that for k jars we can divide our blocks up till $n^{\frac{1}{k}}$. Hence f(n) would be of the order of O($n^{\frac{1}{k}}$) in general.

2 Question 2

We will make use of a variation of Bipartite graphs 2 coloring problem taught in class for this question.

A simple algorithm can be devised as follows:

1. For all m judgments., Perform the following steps:

   Draw a graph such that:

   (a) if labelled same, draw an edge between any 2 nodes (i,j). Mark the edge as 'S'.

   (b) if labelled different, draw an edge between any 2 nodes (i,j). Mark the edge as 'D'.

2. Now, Put in bag node i.

   (a) Take any node i out from the bag.

   (b) For each edge connecting (neighbouring) node i :

      i. If edge is labelled S, then mark the neighbouring node as i. (same as i)

      ii. If edge is labelled D, then mark the neighbouring node as j. (different from i)

3. Put all neighbours of the neighbouring node in the bag.

4. If all vertices are marked then stop, otherwise repeat from step 2.

In the end check whether all nodes are marked with a single label. If any node is marked with two(or more) different contradicting labels ,then it is inconsistent otherwise consistent.

3 Question 3

We have the following notations:

$\mathbf{C}_a$, which is the infected computer.
$\boldsymbol{x}$, time at which C_a was infected.
$\mathbf{C}_b$, Whether it got infected?
$\boldsymbol{y}$, time at which we need to check.
Also we are given with m triples and n computers in total
We can devise a simple algorithm for this problem.

1. Start from time x. End till time y. Mark C_a(initial infected computer).

2. For each triple (C_u C_v t_k) (total m triples) ,

   (a) If C_u or C_v is marked, then mark both.

(b) If t_k is same as previous iteration and C_u or C_v was marked in previous iteration, then mark current C_u and C_v as well. (Handling the case when there are open connections)

3. At end, loop through all the computers. If C_b is unmarked, then return true, else return false. (at the end of time, checking the target computer)

Total running time in this algorithm would be O(n computers) [step 3]+ O(m triples) [step 2]. which is O(m+n).

4 Question 4

We can arrange the given functions in a list which is as follows **(arranged in increasing order:)**

1. $\sum_{i=1}^{n} \frac{i^2+5i}{6i^4+7}$
2. $\sum_{i=1}^{n} \frac{1}{i^2}$
3. $\lg \lg n$ =$\mathbf{O}(\lg(\lg n))$
4. $\sqrt{\lg n} = \mathbf{O}(\sqrt{\lg n})$
5. $lg(\sqrt{n})$=1/2 lg n = **O(lg n)**
6. $\ln n$= O($\frac{\lg n}{\lg e}$) = $\mathbf{O}(\lg n)$ [lg e is a constant]
7. $\sum_{i=1}^{n} \frac{1}{i} = \log \frac{2n+1}{2n-1}$ = **O (log n)** [approx 0.33*lg (n) on calculation]
8. $2^{\sqrt{\lg n}}$= $\mathbf{O}(n^{1/2})$
9. $(\lg n)^{\sqrt{\lg n}} = \mathbf{O}((\lg n)^{\sqrt{\lg n}})$
10. min $(n^2, 1045n) = \mathbf{O}(n)$
11. $\ln(n!) = \ln n^n$ = **O(n log n)**
12. $n^{\ln 4}$= $\mathbf{O}(n^{1.3})$, where ln 4=1.3 approx
13. $\lfloor n^2/45 \rfloor = \mathbf{O}(n^2)$
14. $n^2/45 = \mathbf{O}(n^2)$
15. $\lceil n^2/45 \rceil = \mathbf{O}(n^2)$
16. $5n^3+\log n = \mathbf{O}(n^3)$
17. $\sum_{i=1}^{n} i^{77} = \mathbf{O}(n^{77})$ [max term to calculate is of power 77]
18. $2^{\frac{n}{3}}$= $\mathbf{O}(2^{\frac{n}{3}})$

19. $3^{\frac{n}{2}}=2^{lg3.\frac{n}{2}} \approx 2^{0.8*n}= \mathbf{O}(2^{0.8*n})$ [lg3 is approximately 1.6]

20. $2^n = \mathbf{O}(2^n)$

Here 1) $\equiv$ 2) $\equiv$ 3) $\prec$ 4) $\prec$ 5) $\equiv$ 6) $\equiv$ 7) $\prec$ 8) $\prec$ 9) $\prec$ 10) $\prec$ 11) $\prec$ 12) $\prec$ 13) $\equiv$ 14) $\equiv$ 15 $\prec$ 16 $\prec$ 17) $\prec$ 18) $\equiv$ 19) $\equiv$ 20)

5 Question 5

1. We will prove this by contradiction. We are given Graph G. If otherwise, Suppose any vertex of G has an odd degree. Now when we traverse graph G for a Euler tour (cycle), then for every middle node in graph G, we need to have even degree because we enter any given vertex via some edge and leave the vertex via different edge (each edge travelled only once). If any of the vertices have odd degree (say 3), then there is no way to leave that vertex except going back via the same edge which contradicts the definition of Euler tour. (each edge traversed once)

   Now to prove the other part of the question. if every vertex is even, then prove that it is Euler tour. If every vertex is even, then there will be no open ends (degree 1). The only way that is possible is when we have a close cycle. Hence suppose we start travelling from an initial vertex, then we would need to enter and leave every vertex along the way. Hence degree would become 2 from 0. Finally as we reach the starting vertex, we would need to join that vertex to create a Euler tour which proves the above theorem

   Hence every vertex needs to have even degree for an Euler tour.

2. We can devise the following algorithm for Euler tour:

   (a) Take a node v from graph and put it in the bag.

   (b) while(bag is not empty)

      i. Take any node v out from the bag.
      ii. if degree(v) is odd, then return false [from above proof]
      iii. For any one edge (u,v) connecting from v, if not already marked, then mark (u,v). Put (u,v) in an array euler[].
      iv. choose the other vertex of this connecting edge i.e. u and put it in the bag.

      If all vertices and edges are not finished, then backtrack until you finish.

      Stop when you have finished all the edges and when you reach the initial vertex.

   If all edges are not marked then not Euler. If Euler, print array euler[].

6 Question 6

PROOF: We are given a connected acyclic graph with n vertices. Then there would be n-1 edges by definition of acyclic connected graphs.

Now, since the graph is an open graph (no cycles) , there must be a longest path P from one end to the other. So there must be at least two open ends. Hence there should be at least 2 vertices of degree 1 in acyclic connected graph (the two ends). To get no vertices of degree 1 is to close a cycle, since we would need to join both or one of the ends at some point of time. Which is a contradiction. Hence for acyclic connected graph with n¿=2 we will have at least 2 vertices of degree 1.

7 References

1. http://m.wolframalpha.com/
2. https://math.stackexchange.com/
3. https://www.quora.com/