Homework 3

Alejandro Güereca Valdivia

3/20/2017

1. Exercise 5.18 of the Dasgupta

- What is the expected number of bits per letter?

- C++ implementation of this case using can be found in this link:
  - https://github.com/Dragv/Algorithms/tree/master/Homework2

2. Exercise 15.10 from the Cormen

- Suppose that there exists an optimal solution S that involves investing $d_1$dollars into investment k and $d_2$dollars into investment m in the first year. With this solution you don't move the money for the first j years. We can perform the classic cut-and-paste maneuver if $r_{k1} + r_{k2} + ... + r_{kj} > r_{m1} + r_{m2} + ... + r_{mj}$. Using the cut-and–paste idea we can invest d$d_1 + d_2$into investment k for j years.
- We need to solve two optimal subproblems if the investment strategy is the year-one-plan strategy. The first one is either keeping the strategy for another year or moving the money. Because of this, the problem shows optimal substructure.
- C++ implementation of this case using can be found in this link:
  - https://github.com/Dragv/Algorithms/tree/master/Homework2
- The running time of this algorithm is O(n)

3. Exercise 17.2 from the Cormen

- In this case. We go through the lists in a linear matter and binary search each one since we don't know the relationship between lists. The worst case scenario for this case is if all of the lists are used. List i has a length of 2^iwe can search it in $O(i)$, because it's als sorted. Due to the fact that i varies from 0 to $O(lgn)$ so the runtime of SEARCH is $O((lgn)^2)$.
- With the INSERT operation it's needed to change the first m 1's in a row to 0's, and then change a 0 for a 1 in the binary representation. Then we combine the lists, and insert the new element into the new list created from the combination. In the worst case this takes $O(n)$.
- In the DELETE case. First we need to find the smallest m such that $n_m \neq 0$ in the binary representation n. In the case that the item to be deleted isn't in $A_m$it needs to be removed from its list and swap it with an element from A_m. This can be done in $O(lgn)$. Now we break $A_m$into seperate lists by index. Due to the fact that the lists are already sorted the whole running time lays in the splitting of the lists which takes $O(m)$. The overall running time of the DELETE is $O(lgn)$.