Hermes Espínola González

March 11, 2017

Homework 3

1. From Dasgupta
   (a) Do exercise 5.18 in Chapter 5
   - What is the optimum Huffman encoding of the English language?

Character	Frequency	Encoding
blank	18.3%	111
e	10.2%	010
t	7.7%	1100
a	6.8%	1010
o	5.9%	1000
i	5.8%	0111
n	5.5%	0110
s	5.1%	0011
h	4.9%	0001
r	4.8%	0000
d	3.5%	10111
l	3.4%	10110
c	2.6%	00101
u	2.4%	00100
m	2.1%	110111
w	1.9%	110101
f	1.8%	110100
g	1.7%	100111
y	1.6%	100110
p	1.6%	100101
b	1.3%	100100
v	0.9%	110110
k	0.6%	11011010
j	0.2%	1101101110
x	0.2%	1101101111
q	0.1%	1101101100
z	0.1%	1101101101

   - What is the expected number of bits per letter? 4.2 bits

(b) Then implement the Huffman's code solution in C. Then, selecting a large enough corpus compare tell me what was your average compression.

2. From Cormen

- Do exercise 15.10 (Back of the chapter 15) and implement the solution in C. Please, run it against the scenario described by http://statmath.wu-wien.ac.at/~zeileis/grunfeld/

   a. Supposing that there exists an optimal solution d_1 dollars into investment i_1 and d_2 dollars into investment i_2, now suppose that we don't move our money for j years. If $r_{i_1,1} + r_{i_1,2} + ... + r_{i_1,j} > r_{i_2,1} + r_{i_2,2} + ... + r_{i_2,j}$ then, by contradiction, there is an optimal investment in which we invest $d_1 + d_2$ in a single investment.

   b. For any given year j the maximum return rate so far is $r_{i{kj},j} + $r_{i_kj-1,j-1} + ... + r_{i_k1,1}$, and the maximum return rate for year j is given when $k_j = max_{1 \leq i \leq n}(ri, j)$ and thus the problem exhibits optimal substructure.

   c. Design an algorithm that plans your optimal investment strategy. What is the running time of your algorithm?

      - Code in folder optimal_investment
      - Because we have a for loop from 1 to n inside a fixed for loop that gets executed exactly 10 times, we can say that the algorithm has a running time of O(n).

   d. When there is a limit in the amount of money you can invest, the amount you have to invest the next year becomes relevant. We are left with the problem of investing a different initial amount of money, so we would have to solve a subproblem for every possible initial amount of money.

3. From Cormen 17-2

   a. We do binary search on each array. In the worst case we search in every array. Since each array has length 2^i, we can search it in $O(lg(2^i)) = O(i)$. Since i varies from 0 to $O(lgn)$, the running time of SEARCH is $O((lgn)^2)$

   b. We must change a 0 to a 1 in the binary representation of n. We must change the first m 1's in an array to 0's and combine the first m arrays into A_m and insert the new item into A_m. Merging these arrays can be done linearly, thus O(2^m). In the worst case, this takes time O(n) since m could be equal to k. Since there are 2^m items the amortized cost per item is $O(1)$.

   c. Find the smallest m such that $n_m! = 0$ in the binary representation of n, If the item to be removed is not in array A_m, remove it from its array and

swap an item from A_m, arbitrarily. This can be done in $O(logn)$ time since we may need to search lost A_k to find the element to be deleted. Now simply split the array into arrays $A_0, A_1, ..., A_{m-1}$. This takes $O(m)$ time. In the worst case this is \$O(lgn).