CS3230 Assignment 9

Zhuang Jianning
A0214561M

1. $A = \{a_1, a_2 \dots a_n\}$ where $a_1 < a_2 < \dots a_n$

A and B are sets of n distinct positive integers each

$B = \{b_1, b_2, \dots b_n\}$ after reordering

Find a reordering of B to maximize product $\prod_{i=1}^{n} a_i^{b_i}$

a) Claim. product $\prod_{i=1}^{n} a_i^{b_i}$ is maximized when B is also sorted in ascending order $b_1 < b_2 < \dots b_n$

Proof. By contradiction

Suppose there exists some $i<j$ such that $a_i < a_j$ but $b_i > b_j$ which maximizes product

considering the contributions to the product at indices i and j, $a_i^{b_i} a_j^{b_j}$

$$\frac{a_i^{b_j} a_j^{b_i}}{a_i^{b_i} a_j^{b_j}} = \left(\frac{a_j}{a_i}\right)^{b_i - b_j} > 1 \quad \text{since } \frac{a_j}{a_i} > 1 \text{ and } b_i - b_j \geq 1$$

Thus $a_i^{b_j} a_j^{b_i} > a_i^{b_i} a_j^{b_j}$ and would result in a greater product if the order of b_i and b_j were swapped $\Rightarrow$ contradiction

$\therefore$ there does not exist some $i<j$ such that $a_i < a_j$ but $b_i > b_j$ which maximizes product

B is sorted in ascending order.

```
MAX_ORDERING_B (B)
    merge_sort(B)
return (B)
```

sort B in ascending order using merge sort

time complexity $= O(n \lg n)$

b)

Modelling a comparison based algorithm as a binary decision tree where each node is a comparison, each branch is the outcome of the comparison and each leaf is a ordering.

There are $n!$ possible orderings / permutations $\Rightarrow$ decision tree $\geq n!$ leaves

$$\text{minimum height } h \geq \lg(n!)$$
$$\geq \lg\left(\frac{n}{e}\right)^n$$
$$= \Omega(n\lg n)$$

height / length of path to leaf determines # comparisons, hence running time

lower bound on running time of comparison based sorting algorithm is $\Omega(n\lg n)$

for all comparison based sorting algorithms with running time $f(n)$

$$f(n) = \Omega(n\lg n) \Rightarrow \lim_{n\to\infty}\left(\frac{f(n)}{n\lg n}\right) > 0 \Rightarrow \lim_{n\to\infty}\left(\frac{f(n)}{n\lg n}\right) \neq 0$$

$$\Rightarrow f(n) \neq o(n\lg n)$$

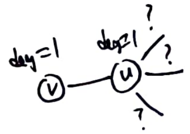
There does not exist a comparison based algorithm running in $o(n\lg n)$

OR use $\exists c, n_0$ argument ?

2. a) independent set $S \Rightarrow$ no edge between any pair of vertices in S

Suppose v is a leaf node in G, maximum independent set contains v

Proof.



Suppose there is an optimal solution T that does not contain v.

Let u be the only neighbor of v (leaf node of degree 1)

case 1: u not chosen, including v would not violate independent set

T is not optimal since there exist a maximum solution with 1 more vertex

case 2: u is chosen. u has degree ≥ 1 since it at least has v as neighbour

replacing u with v in optimal solution T does not decrease size of independent set. Hence maximal independent set contains v.

b) optimal substructure

Suppose S is any optimal solution that contains v

claim: $S - \{v\}$ is optimal for the subproblem with v and all its neighbours removed

Proof. Subproblem without v and neighbours is made up of 1 or more tree.

If T is optimal for subproblem and $|T| > |S - \{v\}|$

Then adding v to T which does not violate independent property since all neighbours were not considered would $|T \cup \{v\}| > |S|$

contradiction

Greedy MIS (G)

$S = \emptyset$

while G not empty

greedily add node with minimum degree to S

remove all neighbours from G

return S

3. coin changing problem for n cents with denominations

$d_1 = 1,\ d_2 = c\ \cdots\ d_k = c^{k-1}$ for $c > 1$ and $k \geq 1$

minimize number of coins used.

a) Greedy Algorithm $\Rightarrow$ Greedily pick largest denomination with value $\leq$ current amount of change

let x_i be the number of coins used for each denomination $d_i = c^{i-1}$

$$x_k = \left\lfloor \frac{n}{c^{k-1}} \right\rfloor$$

$$x_i = \left\lfloor \frac{n \bmod c^i}{c^{i-1}} \right\rfloor \quad \text{for } i < k \quad \Rightarrow \frac{n \bmod c^i}{c^{i-1}} < \frac{c^i}{c^{i-1}} = c$$

$$\Rightarrow x_i < c$$

number of coins used $= \sum_{i=1}^{k} x_i$ is minimal / optimal

Proof. suppose there exists $x_i' < x_i$ for some $x_i' \geq 1$ that uses less coins

since $x_i < c$ for all $i < k$,

* $i \neq k$ since there are no higher denomination than d_k to change to

$$\sum_{j=1}^{i} x_j c^{j-1} < \sum_{j=1}^{i} (c-1) c^{j-1}$$

$$< c^i - c^{i-1} + c^{i-1} - c^{i-2} + \cdots + c - 1$$

$$< c^i$$

Hence, there is not enough value to exchange all lower denomination coins for a higher one. The only possible way to account for $x_i' < x_i$ is to increase number of lower denomination coins

decreasing x_i by 1 increases x_{i-1} by c, thus # coins increases by at least $c-1$

since $c > 1$, $c - 1 > 0$, total # coins increases $\Rightarrow$ contradiction

$\therefore$ current allocation for x_i is optimal

b) denominations $\{1, 3, 4\}$ for $n = 6$ returns $x_1 = 2,\ x_2 = 0,\ x_3 = 1$ } total = 3 coins

optimal solution is $x_2 = 2$ $\Rightarrow$ total = 2 coins