Week 2

Greedy Algorithm

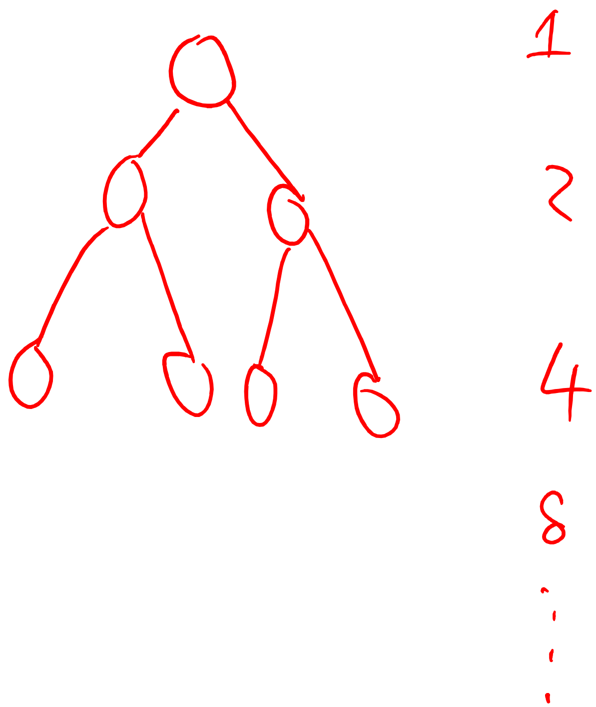
Recall

- Divide and conquer

1) Binary Search
2) Merge Sort
3) Powering a number
4) Multiplying 2 numbers
5) Matrix Multiplication
6) Embedding a complete binary tree into a circuit

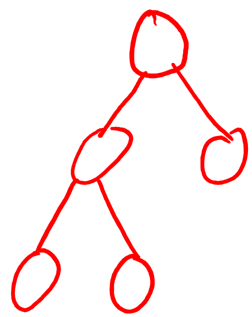
Complete binary tree

- Define: A complete binary tree is a binary tree is a binary tree is a binary tree that has 2^n nodes in level i (provided root is at level 0)



Full binary tree

- A full binary tree is called full if every internal node has two children



→ full, not complete

Fact: A complete binary tree is full, but not vise versa

7) Fibonacci Sequence

- $F_1 = 1$
- $F_2 = 1$
- $F_{n+1} = F_n + F_{n-1}, n > 2$

1, 1, 2, 3, 5, 8, 13, 21, 34, …

Q: How to compute the n^{th} Fibonacci number?

$f_1 = 1, f_2 = 1$
for $i = 3 : n$
$\quad f_i = f_{i-1} + f_{i-2}$
return f_n

$O(n)$

F_{15}

F_{20}

Bottom-up Approach

$F[0] = 1$

$F[1] = 1$

for $i = 2{:}\, n$

$\qquad F[i] = F[i-1] + F[i]$

Mathematical Formula

$$F_n = \left\lfloor \frac{\phi^n}{\sqrt{5}} \right\rceil \longrightarrow O(\lg n)$$

$$\phi = \frac{1+\sqrt{5}}{2} \text{ — irrational}$$

Claim: for every $n \geq 2$

$$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

powering a number

$O(\lg n)$

Strassen's Algorithm: $A_{n \times n} \times B_{n \times n}$

Prove the claim by induction

Base Case: $n=2$

$$LHS = \begin{pmatrix} F_3 & F_2 \\ F_2 & F_1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

$$RHS = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

Prove the claim by induction(continued)

Ind Hyp: Assume the result holds for n_0;

i.e. $\begin{pmatrix} F_{n_0+1} & F_{n_0} \\ F_{n_0} & F_{n_0-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n_0}$

Ind Step: Prove it for n_0+1, i.e.

$$\begin{pmatrix} F_{n_0+2} & F_{n_0+1} \\ F_{n_0+1} & F_{n_0} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n_0+1}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n_0+1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n_0} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \underset{\text{(by ind hyp)}}{=} \begin{pmatrix} F_{n_0+1} & F_{n_0} \\ F_{n_0} & F_{n_0-1} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_{n_0+1}+F_{n_0} & F_{n_0+1} \\ F_{n_0}+F_{n_0-1} & F_{n_0} \end{pmatrix}$$

$$= \begin{pmatrix} F_{n_0+2} & F_{n_0+1} \\ F_{n_0+1} & F_{n_0} \end{pmatrix} = \text{LHS}$$

Claim: Every horse in the world if of the same color

Proof: By induction on the number of horses, say n

Base Case: $n = 1$ ✓

Inductive Hypothesis:

Assume the result holds for n_0 horses(i.e. any set of n_0 horses has the same color)

Inductive Step:

Show that every set of $n_0 + 1$ horses has the same color

no color: d

has color $c \neq d$

$\Rightarrow c = d$

II. Greedy Algorithm

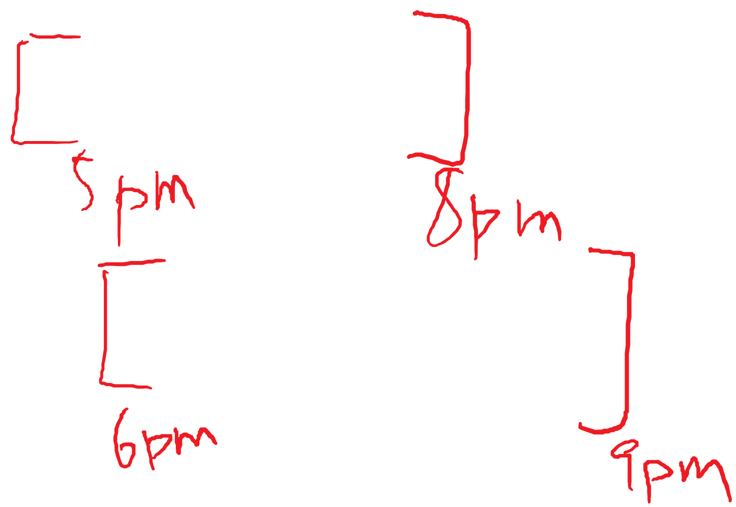
- Newton-Raphson Method
  Hill climbing Method

Example 1: Interval Scheduling

Given: A set $R = \{r_1, r_2, \ldots, r_n\}$ of requests, each with a starting time b_{r_i} and an ending time t_{r_i}

Goal: Schedule the maximum number of jobs possible

(There is only one resource)



Greedy Approach

Define sample rule to decide which request to pick next

Sample rule I:

Pick the job that has the smallest duration (i.e. $t_{r_i} - s_{r_i}$) is minimum

Sample rule II:

Pick the job that starts earliest

Sample rule III:

Pick the job that ends earlist

Generic Approaches

I. “Stay Ahead Approach”
II. “Exchange Argument Approach”
III. “Lower Bound” Approach

Algorithm

Sort the requests R by their end time

Set $A = empty$

while R is not $empty$

    Pick the 1^{st} element $r \in R$

    Add r to A

    Remove r $\not\leqslant$ every other request in R that overlaps with r

end while

return A

Proof

Let θ be an optimal solution

Want to show: $|A| = |\theta|$

$$A = \{i_1, i_2, \dots, i_k\} \;\; |A| = k$$

$$\theta = \{j_1, j_2, \dots, j_m\} \;\; |\theta| = m$$

Claim 1

- For $s = 1, \dots, k$, $t_{i_s} \leq t_{j_s}$

Proof: (by induction on s)

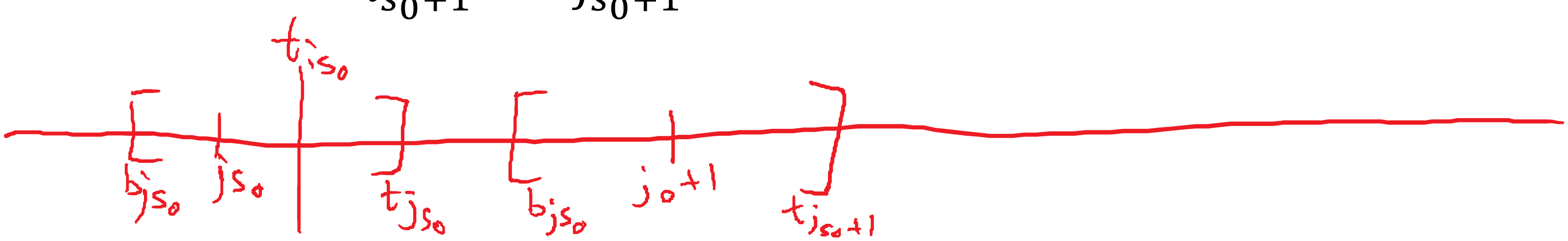
Base Case: $s = 1$

By the way, the greedy algorithm picked the first job, $t_{i_1} \leq t_{j_1}$

Inductive Hypothesis: Assumer the result holds for s_0, $t_{i_{s_0}} \leq t_{j_{s_0}}$

Inductive Step: $s_0 + 1$

To show: $t_{i_{s_0+1}} \leq t_{j_{s_0+1}}$



Claim 1 (continued)

The next job that the greed algorithm picks is the one that ends soonest among the remaining non-overlapping requests

j_{s_0+1} is one such request

is non-overlapping with i_{s_0}

Claim 2

- Greedy Algorithm gives an optimal solution

Proof: $A = \{i_1, i_2, \dots, i_k\}$

$|A| = k$

$\theta = \{j_1, j_2, \dots, j_k\}$

$|\theta| = m$

WTS: $k = m$

Claim 2 (continued)

Proof by contradiction

Assume $k \neq m \Rightarrow m > k$
Apply Claim 1 to $s = k$
We know $t_{i_k} \leq t_{j_k}$

Consider j_{k+1}

So, j_{k+1} is non-overlapping with i_k, and therefore, $j_{k+1} \in R$ when i_k and its overlapping requests are removed from R

$\Rightarrow R \neq \phi$

Weighted Interval Scheduling

- Every job has a priority/weight

Schedule All Intervals

- Multiple resources
- Goal: to schedule all requests using as few resources as possible (KT)

Example 2: Huffman Coding and Data Compression

- Text in some alphabet

$S = \{a, b, c, d, e\}$

$aaebccae$

Goal: Encode the text in minimum number of bits

S has 32 letters

$\rightarrow$ So use five bits

$(32 = 2^5)$

$a \rightarrow 00000$

$b \rightarrow 00001$

$c \rightarrow 00010$

$\vdots$

$\rightarrow$ Fixed Length Encoding

Text: 8 letters

$\text{Total number of bits} = 8 \times 5 = 40$

Variable Length Encoding

$a \to 0$

$b \to 10$

$c \to 01$

$n \to 00001111$

00000 | 00001

a | b

0 | 1

ab | c

Frequency of Letters in the text

- $x \in S$
- $f_x =$ function of times x occurs in the text

Let the size of the text be n

Then x occurs nf_x times

Let $\gamma: S \rightarrow$ set of codes

$\gamma(n) =$ code of x

Total length of encoding of the text $= \sum_{x \in S} nf_x \cdot |\gamma(x)|$

Average number of bits per letter

$ABL(\gamma) = \sum_{x \in S} f_x \cdot |\gamma(x)|$

Frequency of Letters in the text (continued)

Goal: Come up with an encoding γ such that $ABL(\gamma)$ is minimum possible

ab→010
ac→011
bc→1011

$a \rightarrow 0$ 0

$b \rightarrow 1$

$c \rightarrow -1$ 11

a is part(prefix) of c

0 | 1

Morse Code

57 — 100 2

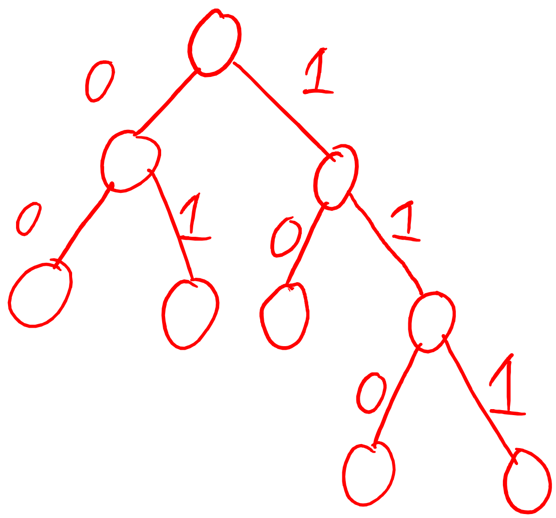
0 pulse 1

ab/2

0 | |

Define: A prefix code γ for S is an encoding such that for any two letter $x, y \in S$, $\gamma(x)$ is not a prefix of $\gamma(y)$

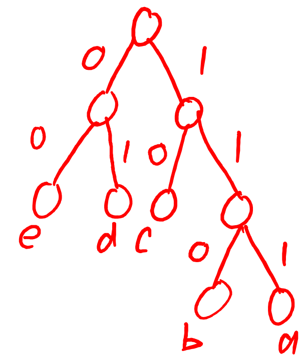
Binary Tree



$e \to 0$
$d \to 01$

d is a prefix of e

$a \to 111$
$b \to 110$
$c \to 10$
$d \to 01$
$e \to 00$



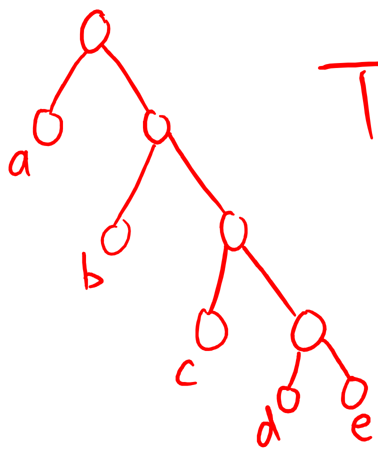
00

Notes

frequency

$a \to 0.4$
$b \to 0.3$
$c \to 0.2$
$d \to 0.049$
$e \to 0.051$

$\underline{a \to 0}$



T^*

Let T^* is an optimal tree

Claim: Suppose u of v are two leaves of T^*

leaf u is labeled with x, and leaf v is labeled with y

$Depth(u) < Depth(v)$

Then, $f_x \geq f_y$

Algorithm:

if S has 2 letters x and y

then $x \to 0, y \to 1$

else

Let y^* and z^* be two lowest frequency letters

Form a new letter w and $S' \coloneqq S \cup \{w\} \setminus \{y^*, z^*\}$ with $f_w \coloneqq f_{y^*} + f_{z^*}$

Recursively construct a prefix code γ' for s' with tree T'

Define a prefix code for S as follows:

Start with T'

Take the leaf labelled w

Add 2 children to that node,label them as y^* and z^*

End if

Notes

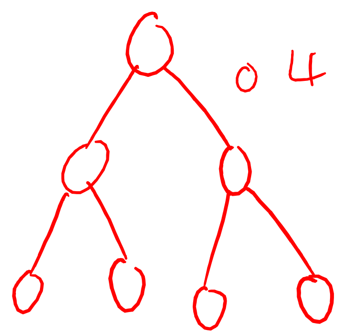
a → 0.4

b → 0.3

c → 0.2

d → 0.05

e → 0.05



a	0	4
b	0	3
u	0	3