★ Inverse FFT

$C(x) = A(x) \cdot B(x)$

$\therefore C(\omega_{j,2n}) = \sum_{i=0}^{2n-1} c_i (\omega_{j,2n})^i$

$$\begin{bmatrix} 1 & \omega_{0,2n} & \omega_{0,2n}^2 & \cdots & \omega_{0,2n}^{2n-1} \\ 1 & & & & \vdots \\ \vdots & & & & \\ 1 & \omega_{2n-1,2n} & \cdots & & \omega_{2n-1,2n}^{2n-1} \end{bmatrix} = V$$

(Vandermonde matrix)

★ $V_{jk} = \omega_{j,2n}^{k} = \omega_{jk,2n}$

$V_{jk}^{-1} = \frac{-\omega_{jk,2n}}{2n}$ } Claim

★ We know $VV^{-1} = I = V^{-1}V$

i.e. $[VV^{-1}]_{ii} = 1$

$[VV^{-1}]_{ij} = 0 \quad (i \neq j)$

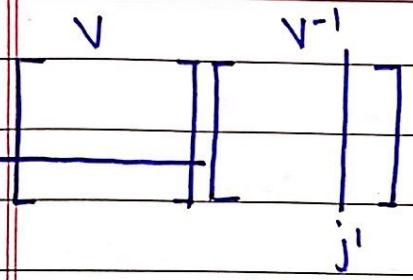
★ Lemma: For $m \geq 1$ and $k \in \mathbb{Z}^+$ which is not a multiple of m, $\sum_{j=0}^{m-1} (\omega_{k,m})^j = 0$

Proof: $\sum_{j=0}^{m-1} (\omega_{k,m})^j = \frac{(\omega_{k,m})^m - 1}{\omega_{k,m} - 1}$

$\because$ k is not a multiple of m $\therefore$ denominator is not zero Hence $(\omega_{k,m})^m - 1 = 0 \Rightarrow$ Given sum = 0

↓

mth root of unity



$$[VV^{-1}]_{jj'} = \sum_{k=0}^{2n-1} V_{jk} \cdot V^{-1}{}_{kj'}$$

$$= \sum_{k=0}^{2n-1} w_{jk,2n} \cdot \frac{w_{jk,2n}}{2n}$$

$$= \frac{1}{2n} \sum_{k=0}^{2n-1} (w_{k,2n})^{j-j'}$$

∴ If $j = j'$ Then $[VV^{-1}]_{jj'} = 1$

If $j \neq j'$

$$[VV^{-1}]_{jj'} = \frac{1}{2n} \sum_{k=0}^{2n-1} (w_{j-j',2n})^k$$

★ Hence By using the lemma $\because 2n \nmid j-j'$

$\therefore [VV^{-1}]_{jj'} = 0$

$$\therefore V^{-1}{}_{jk} = -\frac{w_{jk,2n}}{2n}$$

★ $$y_k = \sum_{j=0}^{2n-1} a_j \cdot w_{kj,2n}$$

★ $$c_k = \sum_{j=0}^{2n-1} y_j \cdot -w_{kj,2n} / 2n$$ } Can be calculated using FFT (inverted version) in $O(n \log n)$

★ Repeated Squaring

→ Modular exponentiation

$$x^y \bmod n = \begin{cases} (x^{\lfloor y/2 \rfloor} \bmod n)(x^{\lfloor y/2 \rfloor} \bmod n) \bmod n & \text{If } y \text{ is even} \\ (x^{\lfloor y/2 \rfloor} \bmod n)(x^{\lfloor y/2 \rfloor} \bmod n) x \bmod n & \text{If } y \text{ is odd} \end{cases}$$

No. of recursive calls : $O(\log y)$

T.C = $O(\log y \cdot \max(\log x, \log y, \log n))$

↓

RAM model

★ $Y = Y_0 \text{LSB}$

$X^Y = X^{Y_0 \text{LSB}}$

If LSB = 0 then Y is even, else Y is odd

★

```
int power(int x, int y, int n)
{
  int res = 1;
  x %= n
  while(y) {
   if(y & 1)
        res=(res*x)%n;
   y>>=1;
   x=(x*x)%n;}
  return res;
}
```

* Greedy Algorithm

① JOB SCHEDULING

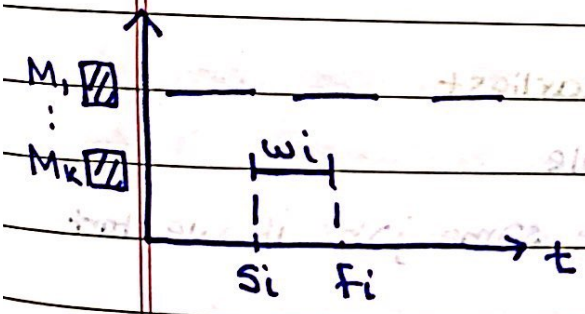
Homogeneous machines : same performance

Heterogeneous machines: different performance

* **No preemption:** Machine is occupied from the start time of the job till the job ends.

* **Mutual Compatibility:** No overlapping in jobs

→ Time interval is contiguous.



|S| = n

K: machines

wi : profit with ith job

* Optimal Solution:

① Cast the optimization problem as one in which we make a choice and are left with one subproblem to solve.

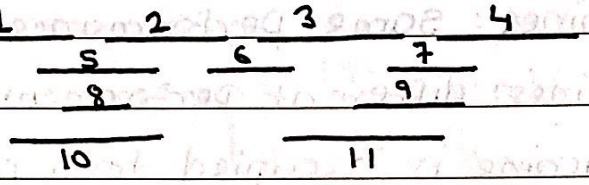
② Check whether it is feasible

③ Check the optimality:

By choosing any subset of S you cannot better the solution/profit

✱ JOB SCHEDULING:

1 Machine, n Jobs (Finish max no. of jobs)



$S^1 = \{1, 2, 3, 4\}$: Optimal & feasible (Optimal Solution)

$S^{11} = \{1, 3, 4\}$: Only feasible

$S^{111} = \{1, 2, 3, \ldots, 11\}$: Optimal but not feasible

S1: Select the job that starts the earliest

_____ : Counter example

Reason to pick: We might lose some jobs if we start late.

S2: Select the shortest job

_____ _____ : Counter example

Reason: Machine will have more free time

S3: Select the job with fewest conflicts:

Counter example: 1st dia on page 11 segment

Reason: Need to remove the fewest no. of jobs.

S4: Select the job with earliest finish time.

Reason: Machine could be freed earlier.

① Maintain two arrays

(one with jobs sorted in finish time order and second with the corresponding start times of array 1)

$f_1\ f_2 \ldots\ldots f_n$ (sorted)

$s_1\ s_2 \ldots\ldots s_n$

Greedy Strategy: Earliest finish time

* Induction on no. of jobs in the output set

→ At every juncture of the algorithm, no compatible jobs get removed and every incompatible job is guaranteed to be removed after placing a job in output array.

Induction Basis:

First job with earliest finish time inserted in the set and pruning is done correctly.

f_1 ✓ f_2 ✗ f_3 ✗ f_4 ✓ f_5 f_6

s_1 s_2 s_3 s_4 s_5 s_6

Let f_1 be the job with earliest finish time and let $s_2, s_3 < f_1$, Hence we cant choose job 2 & job 3 in output set. if $s_4 > f_1$ insert job 4 in output set. Note in case $s_i < f_1$ for $i > 4$ it is anyways going to be less than f_4.

Hence pruned correctly.

Induction Hypothesis:

$S' = \{j_1, j_2, \ldots j_i, \ldots\}$

upto ith job we have correctly choosen (followed greedy strategy)

Inductive Step

Next job would be choosen with earliest finish time

* The solution output by greedy algorithm is feasible i.e. every two jobs in S' are compatible.

Let two jobs of S' not be compatible

say j_p and j_r

WLG let $p < r$ $\therefore$ $f_p > s_r$

Consider q such that $p \leq q < r$

such that: $f_q \geq f_p > s_r$

But when we insert j_q in set S'

we must remove j_r. Hence contradiction

✱ STAYS AHEAD ARGUMENT:

∃ a Optimal Solution # Naive : $\Omega(2^n)$: O

Greedy output : S'

Let n' be the number of jobs in S'. For every $1 <= i <= n'$ i^{th} job choosen into S' stays ahead of i^{th} job in the optimal solution O when jobs in O are sorted according to their finish times.

Induction on no. of jobs selected in S'

Induction Base:

$f_1 \leq f_1'$ (# Greedy Strategy)

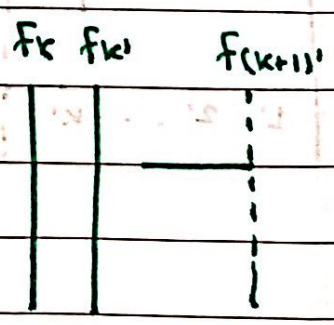
Induction Hypo:

$f_1 \leq f_1'$, $f_2 \leq f_2'$, ... , $f_k \leq f_k'$

Inductive Step:

TPT: $f_{k+1} \leq f_{(k+1)'}$

We can choose $(k+1)^{th}$ job such that $f_{k+1} \leq f_{(k+1)'}$



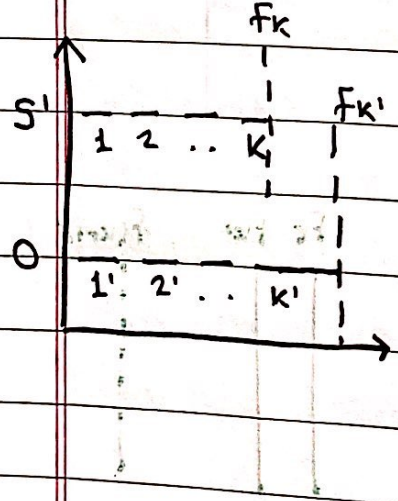
Hence by Induction S' stays ahead of any optimal solution O.

* Given that the solution output by greedy algo S' 'stays ahead' of any optimal solution O, then S' must be an optimal solution.

If $|S'| < |O|$ consider jobs with ids $|S'|+1, \ldots |O|$ in the sorted order of jobs in O, let it be O'

Every job in O' is compatible with every job in S' & hence available for greedy algo after it had choosen set S' of jobs.

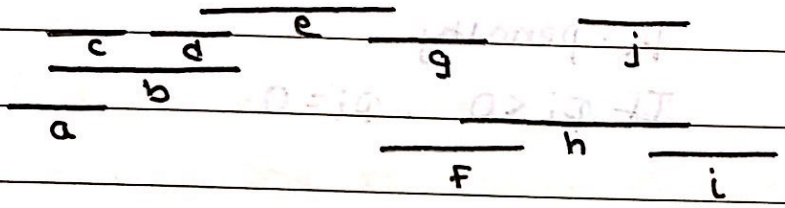
* Note greedy algo stops only when there are no compatible jobs to the ones which are already chose



Date
Page

* Find Minimum no. of machines required to schedule all the tasks.

→ Duration of n tasks given



* Pool of machines : S

Assign a machine at the start of the task

If no machine among the pool is available push a new machine to the pool.

As soon as a task is completed the machine performing that task is free.

* Hence according to input instance any algo should use atleast the maximum no. of conflicting machines.

* MEETING THE LOWER BOUND:

Input instance says any algo should use this much amount of minimum resources and our algo matchs it.

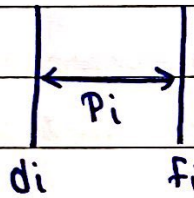
→ Algo: Push all start times & finish times in a vector and sort according to their values & maintain counter for no. of overlaps.

O(nlogn)

* Deadlines

Minimise the maximum penalty

Each task given length & deadline



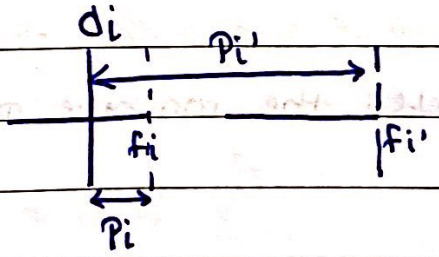
Pi : penalty

If $p_i < 0$, $p_i = 0$

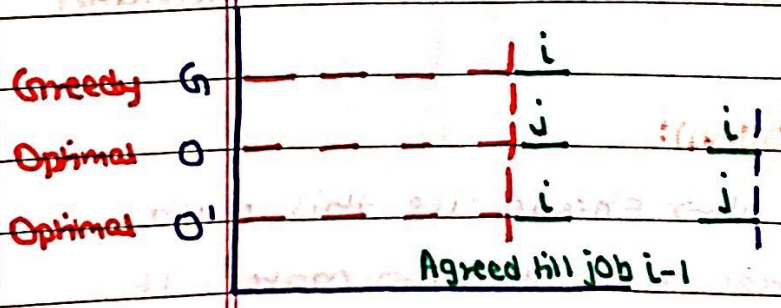
* GREEDY STRATEGY:

Earliest deadline first

Reason: More free time to the machine



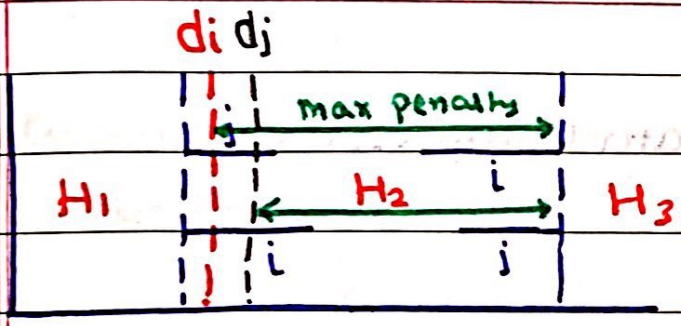
* EXCHANGE ARGUMENT:



O' agrees with G till ith job.

$d_j > d_i$ (# Greedy Strategy)

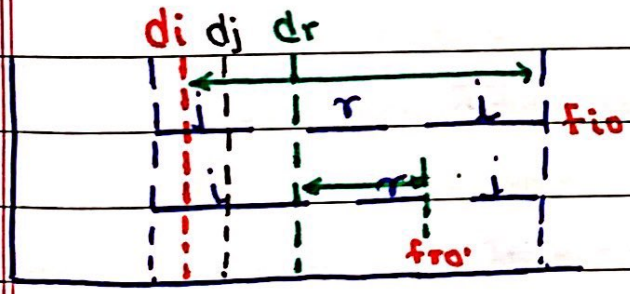
* Optimal solution exists can be found in O(n!)



∴ max penalty (O') ≤ max penalty (O)

* If interval r ∈ H_1
then we have no change in its penalty
or so for r ∈ H_3

* Let r ∈ H_2



dr > di
(# O' follows Greedy Algorithm)

and $f_{io} > f_{ro'}$

∴ max penalty (O') ≤ max penalty (O)

* Hence we keep on making optimal solution more coherent with the greedy solution hence our Greedy solution is optimal.

* Huffman Codes and Data Compression.

$S = \{a, b, \ldots, z\}$

(Text) T → Symbols from S

* Also freq of each symbol is given

a b c .... z

f_a f_b f_c .... f_z

γ : Code

$\gamma(\alpha)$: Codeword

* Fixed length code

$> 2^6$ $< 2^7$ symbols

7 bits for each symbol

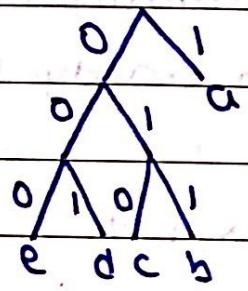
* Avg bits for symbol $= \sum_{x \in S} f_x |\gamma(x)|$

* Prefix codes

No code is prefix of any other code.

a:11 b:01 c:001 d:10 e:000

* Given a binary tree + leaves associated with symbols
→ prefix code



a : 1 $f_a = 0.32$

b : 011 $f_b = 0.25$

c : 010 $f_c = 0.20$

d : 001 $f_d = 0.18$

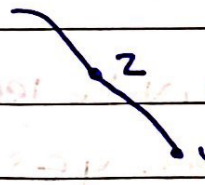
e : 000 $f_e = 0.05$

→ Proof by contradiction

Let $\gamma(z)$ be prefix of $\gamma(y)$ where $z \neq y$

∴ $\gamma(z)$ is prefix of $\gamma(y)$

⇒ z is some node in the path from root to the leaf node y



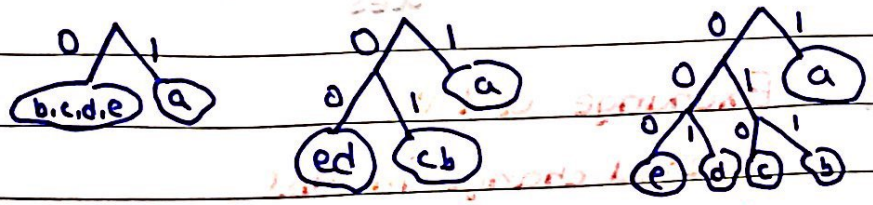
But z must be a leaf node

∴ z = y

to $z \neq y$

* Prefix code given can construct the tree

a:1 b:011 c:010 d:001 e:000



* Binary Tree corresponding to optimal prefix code is full.

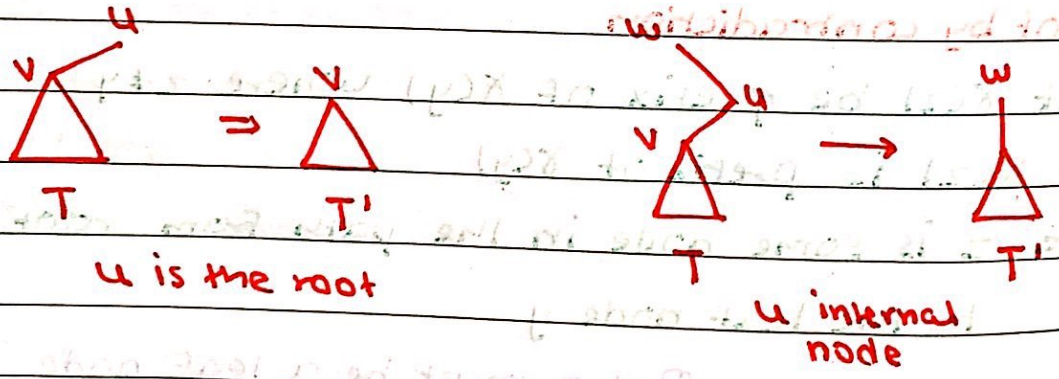
Proof by Exchange Argument

Let T be the binary tree corresponding to optimal prefix code and ∃ a node u with exactly one child v.

Now convert T → T' by replacing u & v

∴ T' has smaller avg no. of bits.

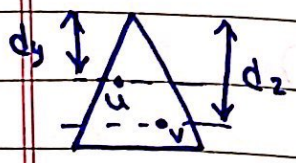
Contradicts optimality of T



* γ^* optimal corresponds to T^* & u, v be leaves ∋ depth(u) < depth(v) If u corres. $y \in S$ & v corres. to $z \in S$ then $f_y \geq f_z$

Proof by Exchange argument

Avg Bits per symbol (ABS) = $\sum_{x \in S} f_x \, \text{depth}(x)$



Exchange u & v

Overall change in ABS

i.e. $\Delta ABS = (\text{depth}(v) - \text{depth}(u))(f_y - f_z)$

If $f_z > f_y$ ∴ ΔABS -ve # to optimality of T^*.

classmate
Date
Page

* There is an optimal prefix code comes to T^* where two lowest freq letters are assigned to leaves that are siblings in T^*

→ The choice of assignment among leaves at same depth doesn't affect the avg no. of bits per symbol.

* Algo for Optimal prefix code:

Let y^* & z^* be having two lowest freq in S.

parent w^* : meta letter with freq

$$f_{w^*} = f_{y^*} + f_{z^*}$$

Replace y^* & z^* with this w^*

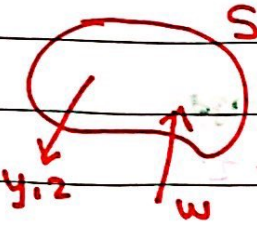
$$ABS(T') = C + f_{w^*} d_{T'}(w^*)$$

$$ABS(T) = C + f_{y^*} d_T(y^*) + f_{z^*} d_T(z^*)$$

$$= C + f_{w^*}(d_{T'}(w^*) + 1)$$

$$\therefore ABS(T) - ABS(T') = f_{y^*} + f_{z^*} = f_{w^*}$$

* Huffman's prefix code



$$S' = S - \{y, z\} \cup \{w\}$$

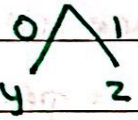
* Binary tree # at every step two symbols removed & hence every node two children (internal node)

* Optimality

→ Induction on no. of symbols

Induction Basis

For 2 symbols



Optimal # only one bit per symbol

Induction Hypo.

$|S| = k-1$ (Optimal)

T.P.T $|S| = k$ also optimal

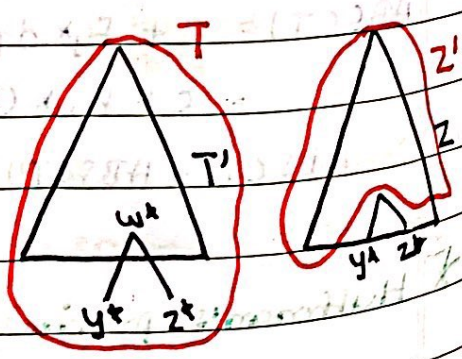
Induction Step

Let S' be set of symbols where $\{y^*, z^*\}$ replaced by w^* which is optimal ∴ I.H.

Let the tree be T'

Attach leaves y^* & z^* to w^*

$ABL(T') = ABL(T) - f_w$



Now let T not be optimal

∴ ∃ Z ∋ $ABS(Z) < ABS(T)$

there is a tree Z ∋ y^*, z^* are siblings

If we delete y^*, z^* from Z to get Z'

∴ $ABS(Z') = ABS(Z) - f_w$

∴ $ABS(Z) < ABS(T)$

⇒ $ABS(Z') < ABS(T')$ # to optimality of T'

* Heap with n symbols
  extract two with min freq
  & insert their sum
  Build O(n)
  Extract O(nlogn)
  Insert O(nlogn)

* ∴ T.C. O(nlogn)

* Greedy Choice Property:
  Many choices choose the one which looks best at the moment.

* Optimal Substructure Property
  Optimal Solution to problem P within it contains optimal substructure (solutions) to subproblems of P.

* Matrix Exponentiation (Repeated Squaring)

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \to f_0 \\ 1 \to f_1 \end{bmatrix}$$

A

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix}$$

Proof By Induction

$$A^n = \begin{cases} A^{\lfloor \frac{n}{2} \rfloor} \cdot A^{\lfloor \frac{n}{2} \rfloor} & \text{if } n \text{ is even} \\ A \cdot A^{\lfloor \frac{n}{2} \rfloor} \cdot A^{\lfloor \frac{n}{2} \rfloor} & \text{if } n \text{ is odd} \end{cases}$$

$T(n) = T(n/2) + O(1)$ if $n > 2$

$= O(1)$ o.w.

$\therefore T(n) = O(\log n)$: Word ram model

$f_n \approx \phi^{n-1}$

$\therefore n \log \phi \approx O(n)$: no. of bits of n^{th} fibonacci no.

Multiply 2 n bit nos

$\therefore T(n) = T(n/2) + M(n)$ $n > 2$

$O(1)$

$M(n) = O(n^{1.59})$ $\therefore T(n) = O(n^{1.59})$