23rd Aug. '18

## Renew

- Start
  - Know the assumptions / limitations
- Understand the obstacles
  - The significant adversary and interference.
- Flow -
  - ~~Fething~~ Techniques like divide & conquer, DP, etc.
- End
  - Be ~~positive~~ proactive

## Assumptions

(Machines are not omniscient)

① Infinite amount of information cannot be stored in finite amount of space.

② (Machines are not omnipresent)
Information travels at a finite speed.

③ (Machines are not omnipotent)

...

* **Turing Machine:**

A turing machine is a 7-tuple $\langle Q, \Sigma, \Gamma, \delta, q_{start}, q_{acc}, q_{rej} \rangle$.

$Q \rightarrow$ finite set of control sets (control unit can be in)

$\Sigma \rightarrow$ finite alphabet set for input.

$\Gamma \rightarrow$ finite tape symbols disjoint set) $(\Gamma \supseteq \Sigma)$

Scanned by CamScanner

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$.

$Q = \{q_0, q_1\}$

$\Gamma = \{a, b\}$

$\delta(q_0, a) = (q_1, b, L)$

$\delta(q_1, b) = (q_0, a, R)$

→ $q_{start} \in Q$ initial state of the machine

→ $q_{acc} \in Q$ accept and halt

→ $q_{rej} \in Q$ ~~halt~~ reject & halt.

⇒ Church - Turing algorithms

An algorithm is a turing machine.

⇛ No turing machine/C program exists for some problem.

- no. of C programs = N.
- no. of computational ~~progr~~ problems = P.

If $P \gg N$, or $P > N$, ~~there~~ proved as there exists atleast one problem that doesn't have any solution.

07/08/18

Q: WAP to ______

= No such program exists.

To prove: No. of programs < No. of problems ⇒
(countable ∞) (non-countable ∞).

* A is countable if
a bijection exists $f : N \rightarrow A$.

* C programs are finite length binary strings. (stored as binary strings.)

$A = \{0,1\}^*$.

$A = \{ \underbrace{\epsilon}, \underbrace{0, 1}_{\text{length } 1}, \underbrace{00, 01, 10, 11}_{\text{length } 2}, 000, \ldots \}$

* Diagonalization technique:

Theorem: $(0,1)$ is uncountable.

Proof = Suppose the contrary.
Let $f$ be a bijection
$f : N \rightarrow (0,1)$.

$f(1) = 0 \cdot d_{11} d_{12} d_{13}$
$f(2) = 0 \cdot d_{21} d_{22} d_{33}$
$f(3) = 0 \cdot d_{31} d_{32} d_{33}$
⋮

To prove: $\exists\ x \in (0,1)$ s.t $\forall\ i \in N \ \{ f(i) \neq x \}$.

$x = 0 \cdot x_1 x_2 x_3 \ldots (x_1 \neq d_{11})\ (x_1 \neq 0 \text{ or } 9)$
$(x_2 \neq d_{22})\ (x_2$ " ——— $)$
$(x_3 \neq d_{33})\ (x_3$ — " ——— $)$
$(x_j \neq d_{jj})\ (x_j$ — " — $)$

$\Rightarrow\ x = 0 \cdot x_1 x_2 x_3 \cdots x_j \cdots$

Hence, we have a contradiction.
So, $(0,1)$ is uncountable.

→ For ~~every programs~~ problems of this type, answer exists in a subset of natural numbers. So, we prove there are $\infty$ such sets then the no. of problems become uncountable.

→ Input : natural no.s
Output : boolean.

We show no. of problems of this kind are itself uncountable.

**Theorem** $P(N)$ is uncountable

**Proof** = Let $f : N \rightarrow P(N)$ be a bijection

$$f(1) = b_{11} b_{12} b_{13} \cdots$$
$$f(2) = b_{21} b_{22} b_{23} \cdots$$
$$f(3) = b_{31} b_{32} b_{33} \cdots$$

Let $S$ is a subset of $N$.

s.t $S = P_1 P_2 P_3 \cdots$ $\quad P_1 \neq b_{11}$, $P_2 \neq b_{22}$

~~$\exists\ i \in N$ s.t $f($~~

So, $\forall\ i \in N$ $f(i) \neq S$, hence $f$ is not bijective

→ (contradiction.)

⇒ No program exists :

1. decidable
   - solve in finite steps
2. undecidable
   - $\infty$ steps / resources
3. unrecognisable
   -

## Problem of 'YES'

Q: WAP to input a C program (M) and its input w, which decides if the answer is Yes.

**Theorem** Problem YES is undecidable.

**Proof:** Suppose some sol$^n$ of YES i.e H exists

$$H(M, w) = \begin{cases} \text{Yes} & \text{if } M(w) \text{ gives} \\ \text{No} & \text{~~else~~ otherwise} \end{cases}$$

→ D on input M
- runs H (M, w)
- if H says yes, says no
- if H says no, says yes

→ what's D (D) ?

Yes → H (D, <D>) = NO ⇓ D(~~<D>~~) ⇒ ~~Yes~~ NO.

No → H (D, <D>) = Yes ⇓ D(D) = yes

---

10$^{th}$ Aug'18

→ Beginnings of Flowing well
- Divide and conquer
- Greedy algo.
- D.P
- Linear Programming

Review

* Starting well (Knowing the assumpt$^n$) Helps
  - in proving impossibility results
  - in "unifying" imp$^n$ problems

→ Multiplication

⇒ (Complex no.s)

$$(a+ib)\cdot(c+id) = (ac-bd) + i(ad+bc)$$

4mult$^n$ ⟶ 3mult$^n$

$P_1 = ac$ $\quad P_3 = (a+d)(b+c)$.

$P_2 = bd$

(Merge, Quick sort → D & C)

(Integer mult$^n$)

⇒ D : $d_{n-1} d_{n-2} \cdots d_2 d_1 d_0 = \sum_{i=0}^{n} (B)^i (d_i)$

E : $e_{n-1} e_{n-2} \cdots e_2 e_1 e_0$

multipl$^n$ $O(n^2)$

$$D = (B)^{n/2} D_L + D_R$$
$$E = (B)^{n/2} E_L + E_R$$

$$D \cdot E = B^n (D_L E_L) + B^{n/2} [D_L E_R + D_R E_L] + D_R E_R$$

$$T(n) = 4T(n/2) + O(n)$$

$\boxed{T(n) = O(n^2)}$ . (Merge sort didn't fail us but in mult$^n$ failed.)

$P_1 = D_L E_L$

$P_2 = D_R E_R$

$P_3 = (D_L + D_R)(E_L + E_R)$

$$DE = B^n \cdot P_1 + P_2 + B^{n/2} (P_3 - P_1 - P_2)$$

$$T(n) = 3T(n/2) + O(n)$$

$$\boxed{T(n) = O(n^{\log_2 3})}$$

(Polynomial mult$^n$)

⇒ $p(x) = \sum_{i=0}^{n-1} p_i (x)^i$

$q(x) = \sum_{i=0}^{n-1} q_i (x)^i$

$\Big\}$ product $p \cdot q(x) = \sum_{i=0}^{2n-1} r_i x^i$

$$r_i = \sum_{k=0}^{i} p_k q_{i-k}$$

(Naive approach $= O(n^2)$)

$$p(x) = p_e(x^2) + x \cdot p_o(x^2)$$

Discrete fourier transform

$a_n a_{n-1} a_{n-2} \cdots a_0 \longrightarrow p(w^0), p(w^1), \ldots, P(w^n)$

$w \rightarrow n^{th}$ root of unity

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^n \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} P(\omega^0) \\ P(\omega) \\ P(\omega^2) \\ \vdots \\ P(\omega^n) \end{bmatrix}$$

$$M_{ij} = (\omega^i)^j$$

17th Aug.'18 * **Greedy Algorithms:**

Our first ex. (Activity Selection Process)

**Input:** Set of n-activities

$A = \{a_0, a_1, \dots, a_n\}$

Each activity has a start time $s_i$ and end time $f_i$

**Output:** Max. sized subset of A that are mutually compatible.

$a_i$ and $a_j$ can be scheduled together iff they do not overlap i.e $f_i \le s_j$ & $f_j \le s_i$

→ Ordered in ascending order of $f_i$

$(f_1 \le f_2 \le f_3 \dots \le f_n)$

**Algorithm**

```
S ← {a1}          l ← last added
l ← 1                 elem. in S.
for i=2 to n
{   if (si ≥ fl)
    {  S ← S ∪ {ai}
       l ← i
    }
} (output: S)
```

**Theorem** Algorithm has greedy choice property

→ Supp. let

$B = \{a_{i_1}, a_{i_2}, a_{i_3} \dots, a_{i_k}\}$.

$f_1 \le f_{i_1}$

$f_{i_1} \le s_{i_2}$ $(i_1 < i_2)$

$B' = (B \setminus \{a_{i_1}\}) \cup \{a_1\}$

$f_1 \le s_{i_2}$

So, we can replace $a_{i_1}$ with $a_1$

Optimum substructure property
→ solving same problem ~~by~~ after first element

$A' = \{ a_i \mid a_i \text{ doesn't overlap with } a_1 \}.$

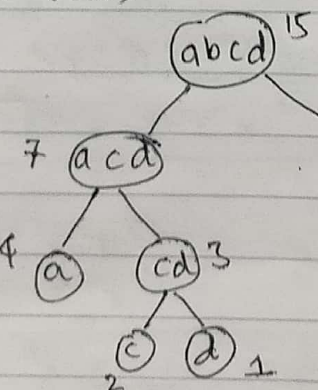
$S = S' \cup \{a_1\}$

→ **Huffman codes**

string abb aaabbcbbbdd b.

| | | |
|---|---|---|
| a | 00 | 10 |
| b | 01 | 0 |
| c | 10 | 110 |
| d | 11 | 111 |
| | 30 bits | 25 bits |

(Huffman tree)

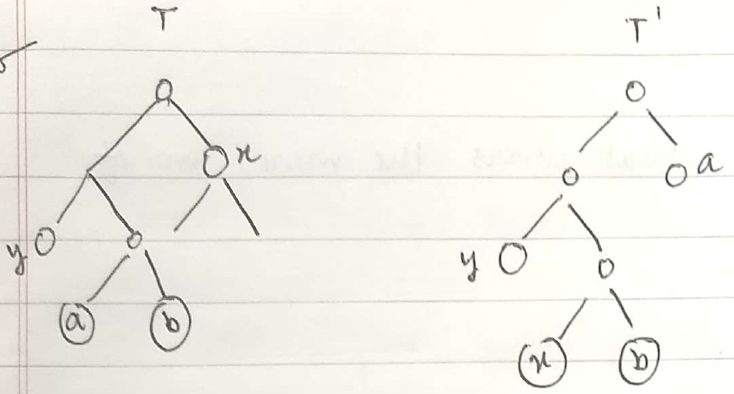



| freq. | | encoding. |
|---|---|---|
| 2 | d | 001 |
| 1 | c | 000 |
| 4 | a | 01 |
| 8 | b | 1 |

$\text{cost} = \sum_{\text{leaf}} f(x)\, d(x)$
f(x) ↑ freq.
d(x) ↑ depth.

Theorem: Huffman codes have the "greedy choice property" i.e if symbols x and y are the two least frequent ones then ∃ an optimal tree with x & y as siblings at max depth.

Proof




~~f(x) d_T(x)~~ to ~~f(a) d~~

$cost(T') - cost(T) \geq 0$ because T is optimum tree.

$= f(x)\, d_T(a) + f(a)\, d_T(x)$

$\quad - f(x)\, d_T(x) - f(a)\, d_T(a)$

$= (\underbrace{f(x) - f(a)}_{\leq 0})(\underbrace{d_T(a) - d_T(x)}_{\geq 0}) \geq 0.$

Either T' is a better sol$^n$ or equally optimum tree as T.

Now, make T'' $(b \Leftrightarrow y)$.

Similarly T'' is an optimal sol$^n$.

---

24th Aug.'18

(Review)
- → Greedy algo
  - Activity selection
  - Huffmann codes.

(Today)
- Matroid theory
- App$^n$ algorithm (set ~~cover~~ cover problem)

= Set Cover Problem:

Set S

(family) $F = \{S_1, S_2, \ldots, S_n\}$.

output: indices $i_1, i_2, \ldots, i_k$ s.t. $\bigcup_{j}^{k} S_j = S$

minimum $k$.

Date ___/___/___

Page ___

myCompanion

Sol$^n$: $I = \Phi$

$U = S$

(uncovered set.) select a set $S_j$ from F that covers the max. no. of elements in U.

$I \leftarrow I \cup \{i_j\}$.

$U \leftarrow U \setminus S_j$

→ Greedy approach may or may not work in this problem.

$|I| \leq O(\log n \, |I^*|)$

= **Matroid:**

set, family

$M = \langle S, F \rangle$ $S \neq \phi$

a) $S \neq \Phi$

b) Heridity property: if $A \in F$, $\forall B \subseteq A$, $\boxed{B \in F}$.

c) Exchange property:

if $A \in F$, $B \in F$

$|A| < |B|$

$\exists\, x \in B \setminus A$. s.t.

$A \cup \{x\} \in F$.

= **Finding a max. weighted independant set:**

$S = \{S_1, S_2, \ldots, S_n\}$ $S_i$ has weightage $S_i, w_i > 0$.

(Greedy works)

If convert any problem to this kind, use greedy.

⇒ All maximal independent ~~size~~ set are of same size.

Date ___/___/___
Page ___

myCOMPANION

: Minimum spanning tree: → subgraph of G (Tree), with all nodes

Input: ~~Undirected~~ Undirected graph (V, E). ($E \neq \phi$).
Edges have (+ve) lengths

Output: Min. weight spanning tree of G.

$M_G = \langle E, F_G \rangle$.

$F_G = \{ A \subseteq E, A \text{ is a cycle} \}$. $(T_A^1, T_A^2)$ $(T_B)$ → spanning across 2 or more trees of A.

~~$S \neq \phi$~~ $(n-|A|) > (n-|B|)$

~~heridity prop^n~~

exchange prop^n. $A \in F_G$ $|A| < |B|$.

$B \in F_G$

acyclic graph → forest.

Theor^n: Any forest with n nodes with t trees has exactly (n−t) edges.

$\boxed{M_G = \langle E, F_G \rangle \text{ is a matroid}}$

use greedy algorithm.

$w_i' = W - w_i$

$E_G = \{ A \subseteq E \mid A \text{ is acyclic} \}$.

---

(Matroid problem)

$M = \langle S, F \rangle$.

$w_i$ = weight of $s_i \in S$, $w_i > 0$ (A is maximal)

$\nexists x$. s.t $A \cup \{x\} \in F$

→ Sort the elements of S in non-decreasing order of weights.

$A = \phi$.

Choose the next $s_i$ (in that order)

  if $A \cup \{s_i\} \in F$.

    $A \leftarrow A \cup \{s_i\}$

$\boxed{\text{output } A}$

$\exists A$ s.t $x \in A$.

Let $B$ be an optimal sol$^n$, $x \in B$ $(B \in F)$

$$S = \{ y \mid y \cup \{x\} \in F \}$$

$$F' = \{ A \in F \mid A \cap \{x\} \in 0 \}$$

$\neq$.