Efficient Algorithm

Algorithm: sequence of steps / atomic operations

such as: $+, -, *, /$

if, for, while, ...

assignments ...

Assumption: every atomic operation can be executed in constant time

Running time of algorithm: # atomic ops executed by alg

This will be instance dependent.

Ex: $(1 || \Sigma_j C_j)$ SPT sorts jobs $j_1, \dots, j_n$ by decreasing processing time: $p_1 \geq \dots \geq p_n$ and schedules them in this order.

Fact sorting n numbers takes $c \cdot n \log n$ steps for some constant c

$\Longrightarrow$ the more jobs an instance has, the more time it takes to run algo

Call an algo A **efficient** if its running time on input I is bounded by some polynomial $p(\text{size}(I))$ of the size of I.

Size: size of I $\approx$ space needed to store I

In the example: need to store n processing times $p_1, \dots, p_n$

Let's assume that numbers are stored in binary format.

Then we need

$$\sum_{j=1}^{n} \lceil \log_2 p_j \rceil$$

bits to store I.

<u>Note:</u> ① $\sum_{j=1}^{n} \lceil \log_2 p_j \rceil \geq n$ (if $p_j > 0 \; \forall j$)

② $\sum_{j=1}^{n} \lceil \log p_j \rceil \geq \lceil \log_2 p_{max} \rceil$

So, $size(I) \geq \max \{ \underbrace{\#\text{input points}}_{n}, \underbrace{\text{space to store data for largest inp. pt}}_{\lceil \log_2 p_{max} \rceil} \}$

An algorithm A for a problem Π is efficient if there is a polynomial p s.t. A takes at most

$$p(\underbrace{size(I)}_{|I|})$$

steps for every instance I.

$\Longrightarrow$ SPT is efficient

<u>Note:</u> running time of SPT is independent of p_{max}

Such algorithms are said to run in strongly-polynomial time.

$\boxed{\text{Asymptotics} \Longrightarrow \text{see slides}}$

Previously you saw an algorithm for Knapsack.

<u>Recall:</u> n items, item j has

profit p_j and weight w_j, param B

Want to choose $S \subseteq \{1, \dots, n\}$ of largest profit,

s.t. $w(S) \leq B$.

You saw an algorithm with running time $O(nB)$.

Is this efficient?

No. Its running time is polynomial in B and not in size(B).

Algs whose running time depends on numbers in input are called pseudo-polynomial.

<u>Reductions</u>

Suppose you needed to convince a friend that obtaining an efficient alg for a problem Π is difficult. How?

Suppose your friend knows Π^0 to be difficult.

<u>One way:</u> if you can solve Π, you can also solve Π^0

Π^0 reduces to Π

Ex: 1) Hamilton Circuit Problem (HCP)

Given graph $G=(V,E)$, is there a Hamiltonian circuit in G?

2) Travelling Salesman Problem (TSP)

Given graph $G=(V,E)$, distances d_{ij} $\forall i,j \in V$, find a tour of min total distance.

Claim: If TSP has an efficient alg, then so does HCP.

Pf: Let A be an efficient alg for TSP. Given an instance of HCP: $G=(V,E)$

let

$$d_{uv} = \begin{cases} 1 & : uv \in E \\ 2 & : \text{othw.} \end{cases}$$

Use A to compute shortest tour T.

- $\sum_{e\in T} d_e = n \Rightarrow$ all edges in T have length 1 $\Rightarrow$ T is a Hamilt. circ.
- T Ham. circ. $\Rightarrow d(T) = n$

Note: Setting up instance takes $O(n^2)$ time. Thus, total running time is polynomial in size(I). ∎

Assume now that we have a blackbox A to solve instances of Π.

If arbitrary instances for Π^0 can be solved in polynomially many comp. steps & polyn. many calls to A, then we say that

Π^0 is polynomial-time reducible to Π : $\Pi^0 \leq_p \Pi$.

In some sense: Π^0 is easier than Π.

$\Longrightarrow$ if Π has a poly-time alg then so does Π^0.

Above claim shows: HCP $\leq_p$ TSP

So, if we knew that no algorithm for HCP existed, then none could exist for TSP either.

Another example:

Partition   n positive integers $a_1, \ldots, a_n$ s.t.

$B = \sum_{i=1}^{n} a_i$ is even

Can you partition $N = \{a_1, \ldots, a_n\}$ into $S \cup T$ s.t. $\sum_{i \in S} a_i = \sum_{i \in T} a_i = B/2$?

Claim: Partition $\leq_p$ Knapsack

Pf: Create the following Knapsack instance:

- Have n objects, object j corresponds to number a_j

- Let $w_j = p_j = a_j \quad \forall j$.
- Choose $\overline{B} = B/2$.

Knapsack size

Let $S \subseteq N$ be optimal Knapsack soln.
If $p(S) = B/2$, then

$\sum_{i \in S} a_i = B/2 \implies$ answer to partition inst. is yes.

Converse similar. ∎

So, in order to convince your friend that solving Π is hard, it suffices to find some known hard problem Π^{o} and prove $\Pi^{o} \leq_p \Pi$.

Unfortunately, we don't know a single such problem Π^{o}!

Here's what we know: we know a huge class of problems s.t.

1. if one has an efficient algo
$\implies$ all of them do
2. none has a known eff. algo

and Partition & HCP are in this class.