CSL 630 Lecture 15 Sept 15

0-1 Knapsack problem

Given items $x_1, x_2 \ldots x_n$ with
weights $w_1\ w_2 \cdots w_n$
profit $p_1\ p_2 \cdot \cdot\ p_n$ and
a knapsack of capacity B, we
want maximize the profit $\sum x_i \cdot p_i$
s.t. $\sum w_i \cdot x_i \leq B$
(B, w_i's are integral)

Let $F(i,j)$ denote the maximum
profit that we can obtain from
objects $x_1\ x_2, \ldots x_i$ and a knapsack
of capacity $j \leq B$
In this notation $F(n, B)$ is the desired
soln

$F(1,1)$ $\begin{cases} \text{if } w_1 \le 1 \text{ then} \\ \quad \text{profit} = p_1 \\ 0 \quad \text{otherwise} \end{cases}$

$\underline{F(1,j)} = \begin{cases} \text{if } w_1 \le j \text{ then } p_1 \\ \text{else } 0 \end{cases}$

$F(2,j)$ if $w_1 + w_2 \le j$ then $p_1 + p_2$

else if . . . – – –

⋮

0

$F(3,j) =$

$F(2,j) = \max$

either soln includes x_2: $p_2 + F(1, \underbrace{j - w_2}_{\geq 0})$

soln doesn't include x_2: $F(1,j)$

$$F(i,j) = \max\{p_i + F(i-1, j-w_i), F(i-1, j)\} \quad (*)$$

$F(i,j)$ ← optimal soln

$1 \le i \le n,\ 0 \le j \le B$

We can iteratively compute

$F(1, j)$ $F(2, j)$ . . . $F(n, -)$
$\downarrow$
(in this ordering) $F(n, B)$ ✓

for all values $0 \le j \le B$

If we store the previously computed term, we need only $O(1)$ to compute $F(i, j)$ from $F(i-1, -)$

$\Rightarrow$ total of $O(\underline{\underline{nB}})$ steps (not 2^n)

If B is bounded by $n^{O(1)}$

$\Rightarrow$ total of polynomial steps

What is the problem size?

$(w_1, w_2, \quad w_n, p_1, p_2, \ldots p_n, B)$

$2n+1$ parameters

size of input is $|w_1| + |w_2| + \cdots |p_1| + \cdots |B|$

So $|B|$ can be half the size of the input

Running time is $2^{|B|} \times n$

Problem : We are given a sequence of n nos

$$x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_n$$

	5,	3,	8,	6,	20,	11,	51,	9
l_i	1	1	2	2	3	3	4	3

An increasing subsequence is

$$x_{i_1} \quad x_{i_2} \quad x_{i_3} \quad \cdots \quad x_{i_k}$$

$$i_1 < i_2 < i_3 \cdots < i_k$$

and $x_{i_1} < x_{i_2} < x_{i_3} \cdots x_{i_k}$

We want to find the longest increasing subsequence.

In any sequence of n nos, there is either an increasing or a decreasing subsequence of length $\lfloor \sqrt{n} \rfloor$

Proof : Consider the longest increasing subsequence ending at x_i and denote it by l_i $\quad 1 \leq l_i \leq n$

If no subsequence is longer than $\sqrt{n}$, some l_i must be repeated at least $\sqrt{n}$

By pigeonhole

$$x_1 \quad \underbrace{\underset{\substack{\downarrow \\ k}}{x_2} \quad x_3 \quad \underset{\substack{\downarrow \\ k}}{x_4} \quad \cdots \quad \underset{\substack{\downarrow \\ k}}{x_i}}_{\text{at least } \sqrt{n} \text{ times}} \quad x_n$$

what can we say about x_2 and x_4

$$x_2 > x_4$$

$\therefore$ There must be a decreasing subs of length $\geq \sqrt{n}$

<u>Erdős - Szekeres thm</u>

If l_i is the longest subsequence ending at x_i

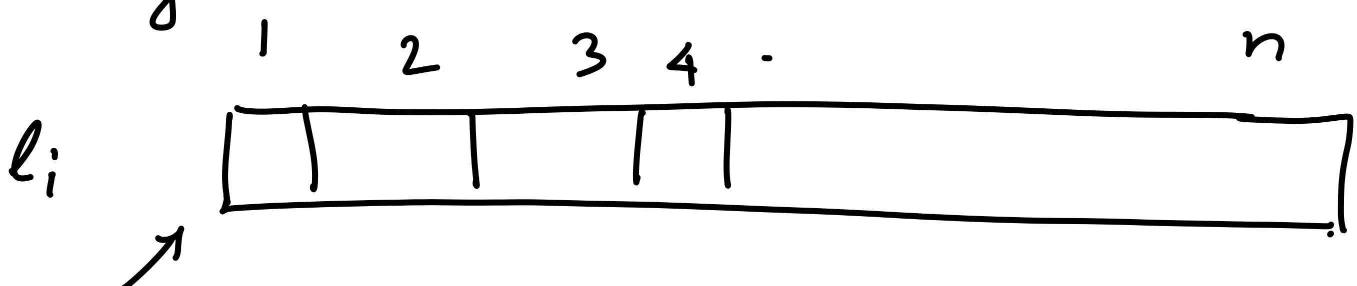
$$l_i = \max_{\substack{j < i, \\ x_j < x_i}} \{ l_j \} + 1$$

l_1 can be computed $l_1 = 1$

To compute each l_i, we need at most $i-1$ lookups

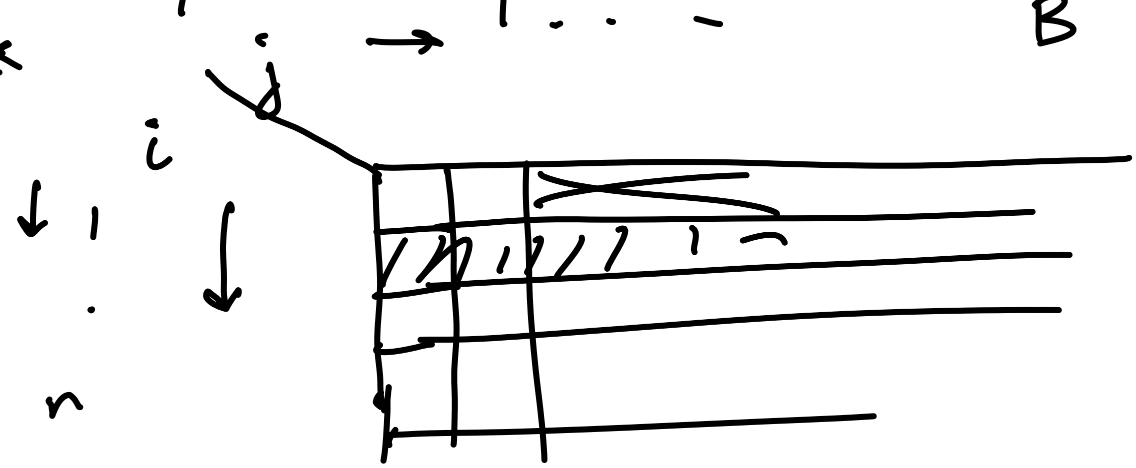
Total time $= \sum_{i=1}^{n} i-1 \sim O(n^2)$

How much space - we need to store all the previously computed l_j



space required

Knapsack



$n \times B$

Faster than $O(n^2)$ soln?

$l_{i,j}$: longest sequence upto i of length $j \leq i$

Moreover, for each l_{ij} we want to retain the sequence that has the smallest last element.