Lecture #11

CS 170

Spring 2021

Dynamic Programming (DP)

General Approach to many problems:

Solve a big problem by breaking it into smaller subproblems, solve subproblems in order from "small" to "large"

Isn't this recursion?

```
func Fib(n)
   if n≤1 return n
   else Fib(n-1)
        +Fib(n-2)
```

cost = $O(\text{Fib}(n)) \sim 1.6^n$

fix - memoization

```
Fib(0)=0, Fib(1)=1
for i=2 to n
   Fib(i)=
    Fib(i-1)+Fib(i-2)
```

cost = $O(n)$

Dynamic Programming - Examples

- Shortest Path in DAG
- Longest Increasing Subsequence

  5, 2, 8, 6, 3, 6, 9, 7 LIS length = 4 vertices

- Edit Distance = fewest # edits to change one string to another (delete, insert, substitute)
  - Spell checking
  - How similar is my DNA to your DNA
  - Cheating detection
- Knapsack: given n items with weights $w_1, \dots, w_n$ values $v_1, \dots, v_n$, total weight limit $= W$

  how to choose subset S of items with $\max \sum_{i \in S} v_i$

  with $\sum_{i \in S} w_i \leq W$

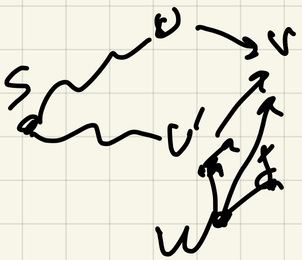
- All-pairs-Shortest-Paths (better than $n \times$ Bellman-Ford)
- TSP - Traveling Salesperson Problem ...

Shortest Paths in DAGs - DP point of view

• Given $G(V,E)$, $w(e) \in \mathbb{Z}$, find shortest path from $s \in V$ to $v \in V$ (negative $w(e)$ ok, no cycles)

• Approach

1) Define subproblems: find shortest path from s to v' for v' "closer to" s than v



2) Show how to solve a problem given solutions to subproblems: $\mathrm{dist}(v) = \min\limits_{u:(u,v)\in E} \mathrm{dist}(u) + w(u,v)$

3) Base case: $\mathrm{dist}(s) = 0$, $\mathrm{dist}(w) = \infty$ if w is a source

4) Choose order to solve subproblems
- topologically sort vertices starting at s

What if we wanted longest path?

$w \to -w$, $\min \to \max$

Longest Increasing Subsequence (LIS)

Given n unsorted numbers $x_1, \dots, x_n$ find LIS

$1, 3, 2, 7, 4, 5, 6$

✗ 1) Subproblems: $f(i) = \text{LIS in } x_1, \dots, x_i$

✗ 2) Solve using subproblems $\quad 5\ 6\ 7\ 1\ 2\ 3\ 4$ oops

1) Subproblems: $L(i) =$ length of LIS in $x_1, \dots, x_i$, ending x_i

2) Solve using subprobs: $L(i) = \max\left(1, \max_{\substack{j<i \\ x_j < x_i}} (L(j)+1)\right)$

3) Base: $L(1) = 1$

5	6	7	1	2	3	4
1	2	3	1	2	3	3+1

4) Order to solve: increasing i

$1\ 3\ 2\ 7\ 4\ 5\ 6$

$$\text{Cost} = \sum_{i=1}^{n} O(i) = O(n^2)$$

Red edges form DAG $x_i \to x_j$ if $x_i < x_j$: Longest Path in DAG

Edit Distance

- How similar are 2 strings $[x_1, \ldots, x_n]$, $[y_1, \ldots, y_m]$?
- How many "edits" needed to change x to y?

  Edit means insert, delete or substitute a char.

snowy	_ s n o w _ y	s _ n o w y	s _ n o w y
sunny	s u n _ _ n y	s u n n _ y	s u n _ n y
	+1 +1 +1 +1 +1	+1 +1 +1	+1 +1 +1
	#edits = 5	#edits = 3	#edits = 3

Motivation: spell-checking - suggest fixes

DNA matching

cheat detection

spam filtering

Edit Distance (ED) between $x=[x_1,\dots,x_n]$ and $y=[y_1,\dots,y_m]$

1) Subproblems: for all $1 \le i \le n$, $1 \le j \le m$

$$f(i,j) = ED([x_1,\dots,x_i],[y_1,\dots,y_j])$$

2) Look at last char in optimal alignment, could be

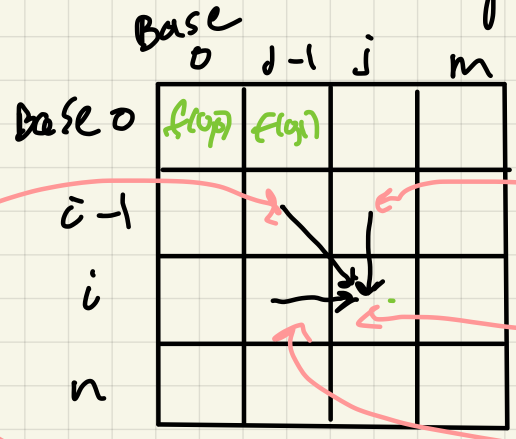
	$\begin{matrix} x_i \\ - \end{matrix}$	$\begin{matrix} - \\ y_j \end{matrix}$	$\begin{matrix} x_i \\ x_j \end{matrix}$
	remove x_i	insert y_j	
Cost =	$f(i-1,j)+1$	$f(i,j-1)+1$	$f(i-1,j-1)+\overbrace{\begin{cases} 0 & \text{if } x_i = y_j \\ 1 & \text{if } x_i \ne y_j \end{cases}}^{d(i,j)=\begin{cases}0 & x_i=x_j \\ 1 & \text{otherwise}\end{cases}}$

$$f(i,j) = \min(f(i-1,j)+1, f(i,j-1)+1, f(i-1,j-1)+\delta(i,j))$$

3) Base case: $f(i,0) = i$ (deletes), $f(0,j) = j$ (insert)

4) Order: for $f(i,j)$ need $f(i-1,j)$, $f(i,j-1)$, $f(i-1,j-1)$

Order of Subproblems for ED



DAG for computing $f(i,j)$

$$f(i,j) = \min\big(f(i-1,j)+1,\ f(i,j-1)+1,\ f(i-1,j-1)+\delta_{ij}\big)$$

x \ y		S	n	o	w	y
	0	1	2	3	4	5
0	0	1	2	3	4	5
S 1	1	0	1	2	3	4
u 2	2	1	1	2	3	4
n 3	3	2	1	2	3	4
n 4	4	3	2	2	3	4
y 5	5	4	3	3	3	3

$\rightarrow$ insert y_j j

$\downarrow$ delete x_i i

$\uparrow$ ED

rowwise, columnwise

$\text{cost} = O(n \cdot m)$

$$\text{Memory} = O\left(\min\begin{pmatrix} m & \text{rowwise} \\ n & \text{columnwise}\end{pmatrix}\right)$$

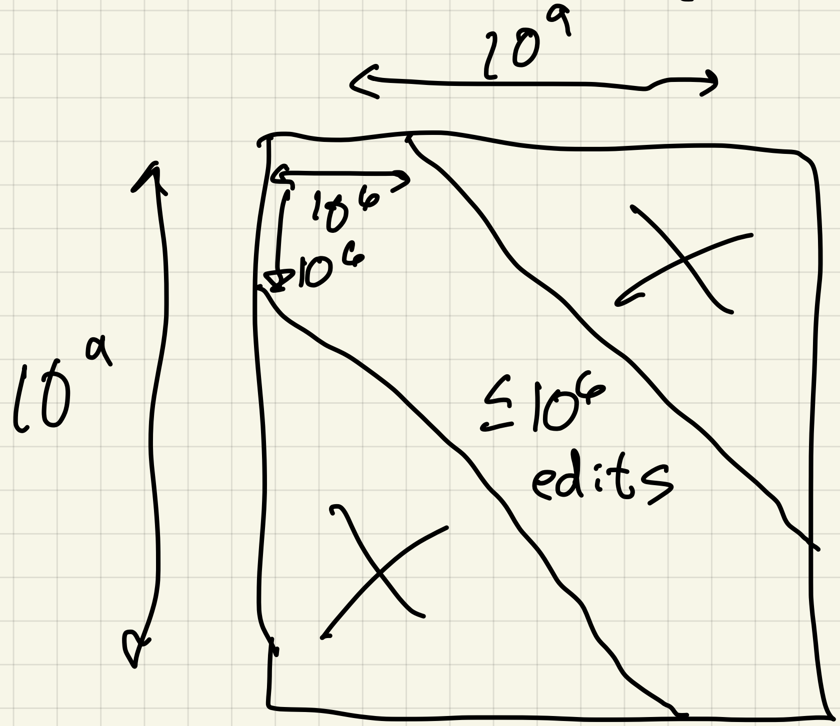
Implicitly shortest path in DAG with edgeweight $\xrightarrow{1}$ $\downarrow 1$ $\searrow \delta_{ij}$

What about $O(m \cdot n)$ cost if $m, n = O(10^9)$?

Why? sequencing DNA

any 2 people have 99.9% same DNA

$\Rightarrow ED = O(.001 \cdot 10^9) = O(10^6)$



only compute $f(i,j)$

for $|i-j| = O(10^6)$

$\Rightarrow$ cost $(10^6 \cdot 10^9) = O(10^{15})$

Smith-Waterman

Needleman-Wunsch

MetaHipMer

Knapsack Problem

- Suppose you are

robbing a jewelry store | deciding how to invest

You have

Knapsack can carry W lbs | \$W to invest

You have to choose among

n jewels, values $v_1, \ldots v_n$ | n investments likely payoffs $v_1, \ldots v_n$

weights $w_1, \ldots w_n$ | costs $w_1 \cdots w_n$

Which should you pick?

$S \in \{1, 2, \ldots n\}$

to maximize $\sum_{i \in S} v_i$ subject to $\sum_{i \in S} w_i \leq W$

Does a greedy algorithm work?

$W = 20$ $\qquad w_1 = 11 \qquad v_1 = 15$

$\qquad\qquad\quad w_2 = 10 \qquad v_2 = 8$

$\qquad\qquad\quad w_3 = 10 \qquad v_3 = 8$

$\left.\begin{matrix} \\ \end{matrix}\right) \begin{matrix} w_2 + w_2 = 20 \\ v_2 + v_3 = 16 \end{matrix}$

Greed: choose i to maximize $v_i \to v_1 = 15$

" " " " $\frac{v_i}{w_i} \to v_1 = 15$

Knapsack by Dynamic Programming

1) Subproblems: $f(i,v)$ = max value packing subset of $1,\dots,i$ max weight $v \le W$

2) Solve $f(i,v)$ = if $w_i > v$ ... w_i too heavy

$$f(i-1,v)$$

else ... could pack w_i

$$\max(\underbrace{f(i-1,v)}_{\text{don't pack } w_i},\ \underbrace{v_i + f(i-1, v-w_i)}_{\text{pack } w_i})$$

3) Base Case: $f(0,v)=0$, $f(i,0)=0$

4) Order: for $i=1$ to n

for $v=1$ to W

$f(i,v) = \dots$ step 2

Cost of Knapsack

for $i=0$ to n, $f(i,0)=0$; for $v=0$ to W, $f(0,v)=0$

for $i=1$ to n
    for $v=1$ to W
        if $w_i > v$
            $f(i,v)=f(i-1,v)$
        else
            $f(i,v)=\max(f(i-1,v), f(i-1,v-w_i)+v_i)$
        end if

$\Theta(1)$

Cost $= O(n \cdot W)$: Polynomial Time?

In "size of input" $v_1, \ldots v_n, w_1 \ldots, w_n, W$

Size of input $O(n(\log_2 \max_i v_i + \log_2 W))$

$n \cdot W$ can be exponentially larger

Knapsack is NP-complete Chap 8