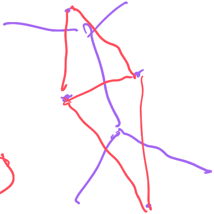
CPSC 420 Lecture 12: Today's announcements:

- HW2 is on Gradescope, due Feb 9, 23:59
- Examlet 2 on Feb 17 in class. Closed book & no notes
- Reading: Ch.H Linear Programming, Ch.3 Dynamic Programming [all by Erickson]

Today's Plan

- Linear programming duality
- Dynamic programming

Linear Programming Duality



Primal

max

min

Dual

$$x_1 + 6x_2 \le 200y_1 + 300y_2 + 400y_3$$

$$\begin{aligned} \text{for all } x_1 &\le 200 \\ x_2 &\le 300 \\ x_1 + x_2 &\le 400 \\ x_1, x_2 &\ge 0 \end{aligned} \qquad \begin{aligned} \text{for all } y_1 + y_3 &\ge 1 \\ y_2 + y_3 &\ge 6 \\ \\ y_1, y_2, y_3 &\ge 0 \end{aligned}$$

The objective value of any feasible solution of the dual LP is an upper bound on the objective value of any feasible solution of the primal LP.

Duality Theorem If LP has bounded optimum then so does its dual and the two optimum values are the same.

Duality in general

$$\max c_1x_1 + \cdots + c_nx_n \qquad \min b_1y_1 + \cdots + b_my_m$$
$$a_{i,1}x_1 + \cdots + a_{i,n}x_n \le b_i \qquad a_{1,j}y_1 + \cdots + a_{m,j}y_m \ge c_j$$
$$x_j \ge 0 \qquad y_i \ge 0$$
$$i = 1\ldots m \quad j = 1\ldots n \qquad i = 1\ldots m \quad j = 1\ldots n$$

matrix notation

max $\vec{c} \cdot \vec{x}$

subject to:

$$A\vec{x} \le \vec{b}$$
$$\vec{x} \ge 0$$

min $\vec{y} \cdot \vec{b}$

subject to:

$$\vec{y}A \ge c$$
$$\vec{y} \ge 0$$

Longest Common Subsequence

What is a longest common subsequence (LCS) of:

A B A N D O N

B A D N O D N O

A string (array of characters) $Z[1..k]$ is a **subsequence** of $X[1..m]$ if there exist indices $i_1 < i_2 < \cdots < i_k$ such that $Z[j] = X[i_j]$ for all $j = 1 \ldots k$.

Given: Two character strings X and Y ($|X| = m$, $|Y| = n$)
Find: LCS of X and Y

Longest Common Subsequence

noncrossing matching in bipartite graph

What is a longest common subsequence (LCS) of:

A B A N D O N

B A D N O D N O N

A string (array of characters) $Z[1..k]$ is a **subsequence** of $X[1..m]$ if there exist indices $i_1 < i_2 < \cdots < i_k$ such that $Z[j] = X[i_j]$ for all $j = 1 \ldots k$.

Given: Two character strings X and Y ($|X| = m$, $|Y| = n$)
Find: LCS of X and Y

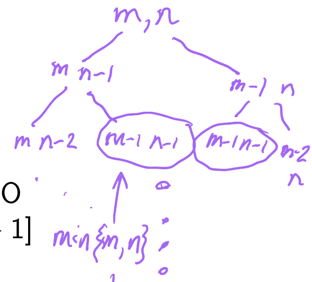
If last characters match (i.e. $X[m] = Y[n]$)
Then there is *some* LCS that ends with this character. (Why?)
So recursively find LCS$(X[1..m-1], Y[1..n-1])$ and add $X[m]$.

Longest Common Subsequence

What if the last characters don't match?

A B A N D O N

B A D N O D N O



Then $X[m]$ might match something in $Y[1..n-1]$
Or $Y[n]$ might match something in $X[1..m-1]$
But not both. (Why?)
So recursivly find LCS($X[1..m], Y[1..n-1]$)
and LCS($X[1..m-1], Y[1..n]$) and return the longest.

worst case runtime $\Omega(2^{\min\{m,n\}})$

LCS(X,Y)
1. $m = |X|$, $n = |Y|$, If $m = 0$ or $n = 0$ return $\emptyset$
2. If $X[m] = Y[n]$ return LCS($X[1..m-1], Y[1..n-1]$)$\circ X[m]$
3. Else return longer of
LCS($X[1..m], Y[1..n-1]$) and LCS($X[1..m-1], Y[1..n]$)

Dynamic Programming

Dynamic programming is a technique for avoiding repeated recursive calls by:

1. Storing the solutions to subproblems.
2. Solving subproblems from the bottom up.

	$\emptyset$	A	B	A	N	D	O	N
$\emptyset$								
B								
A								
D								
N								
O								
D								
N								
O								

Dynamic Programming

Dynamic programming is a technique for avoiding repeated recursive calls by:

1. Storing the solutions to subproblems.
2. Solving subproblems from the bottom up.

length LCS

	∅	A	B	A	N	D	O	N
∅	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
A	0	1	1	2	2	2	2	2
D	0	1	1	2	2	3	3	3
N	0	1	1	2	3	3	3	4
O	0	1	1	2	3	3	4	4
D	0	1	1	2	3	4	4	4
N	0	1	1	2	3	4	4	5
O	0	1	1	2	3	4	5	5

Dynamic Programming LCS

How to store solutions

$$\boxed{T[i,j] = \mathrm{LCS}(X[1..i], Y[1..j])}$$

translate recursion

$$T[i,j] = \begin{cases} T[i-1,j-1] \circ X[i] & \text{if } X[i] = Y[j] \\ \underbrace{\max}_{\text{length}}\{T[i-1,j], T[i,j-1]\} & \text{otherwise} \end{cases}$$

```
LCS(X,Y)
1. T[0,0] = ∅
2. For i = 1 to m  T[i,0] = ∅
3. For j = 1 to n  T[0,j] = ∅
4. For i = 1 to m
5.     For j = 1 to n
6.         if X[i] = Y[j] then T[i,j] = T[i-1,j-1] ∘ X[i]
7.         else T[i,j] = max{T[i-1,j], T[i,j-1]}
8. return T[m,n]
```

empty string

already computed

Running time? $O(m \cdot n)$

already computed

Longest Increasing Subsequence

What is a longest increasing subsequence of:

5 3 4 9 6 2 1 8

A sequence $S[1..k]$ is **increasing** if $S[i] < S[i+1]$ $\forall i = 1..k-1$.

Given: A sequence of numbers $R[1..n]$. Find: LIS of R

Use LCS to solve LIS

LIS(R)
1. S = sort(R)
2. output LCS(S, R)

remove duplicates

$O(n \log n)$

$O(n^2)$

Running time? $O(n^2)$

Faster Longest Increasing Subsequence

To find LIS($R[1..k]$), what information about $R[1..k-1]$ is enough?

A. LIS of $R[1..k-1]$

$$\boxed{\begin{array}{l} 5\ 3\ 4\ 9\ 6\ 2\ 1\ \underset{\uparrow}{8} \\ \qquad\quad\ \ 3\ 4\ 9 \\ \quad\ \ \text{or}\ \ 3\ 4\ 6? \end{array}}$$

Faster Longest Increasing Subsequence

To find LIS($R[1..k]$), what information about $R[1..k-1]$ is enough?

A. LIS of $R[1..k-1]$

5 3 4 9 6 2 1 8
$\uparrow$
3 4 9
or 3 4 6?

B. Best LIS of $R[1..k-1]$

1 2 5 3 4
$\uparrow$
want shorter IS as well

Faster Longest Increasing Subsequence

To find LIS($R[1..k]$), what information about $R[1..k-1]$ is enough?

A. LIS of $R[1..k-1]$

5 3 4 9 6 2 1 8

3 4 9

or 3 4 6?

B. Best LIS of $R[1..k-1]$

1 2 5 3 4

want shorter IS as well

C. Best ISs of length $1, 2, \ldots, j$, where $j = |\mathrm{LIS}(R[1..k-1])|$

Faster Longest Increasing Subsequence

To find LIS($R[1..k]$), what information about $R[1..k-1]$ is enough?

A. LIS of $R[1..k-1]$

$$\begin{array}{l} 5\ 3\ 4\ 9\ 6\ 2\ 1\ 8 \\ \qquad\qquad\qquad\quad\uparrow \\ \qquad\quad 3\ 4\ 9 \\ \text{or}\quad 3\ 4\ 6? \end{array}$$

B. Best LIS of $R[1..k-1]$

$$\begin{array}{l} 1\ 2\ 5\ 3\ 4 \\ \qquad\ \uparrow \\ \text{want shorter IS as well} \end{array}$$

C. Best ISs of length $1, 2, \ldots, j$, where $j = |\text{LIS}(R[1..k-1])|$

Now we need to find best ISs for $R[1..k]$ using this info. How?