

**Test 2:** 7pm next week ! 110mins

**Reading and Exercises** available on the web.

**Links** to summaries from the sibling course at UofT.

*Notes: Different textbook, terminologies, etc.*

## Minimum Spanning Tree Problem

**Input:** A weighted, undirected graph  $G$  (weights on the edges)

**Output:** A minimum-weight spanning tree (or just minimum spanning tree, or MST) of  $G$ .

## Kruskal's algorithm

- Sort the edges in increasing weights:

$$w(e_1) \leq \dots \leq w(e_m)$$

- $E' \leftarrow \emptyset$
- For  $i = 1$  to  $m$  do
- If  $E' \cup \{e_i\}$  does not contain a cycle then

$$E' \leftarrow E' \cup \{e_i\}$$

## Prim's algorithm

1.  $V' \leftarrow \{r\}$  % Start with a node  $r$
2. % Grow  $V'$  by minimum-weight edges:  
While  $|V'| < n$  do
3. Let  $e = \langle u, v \rangle =$  a minimum-weight edge  
crossing  $V', V - V'$  ( $u \in V', v \in V - V'$ )
4.  $V' \leftarrow V' \cup \{v\}$

## GenericMST: Generic algorithm for MST

- $A \leftarrow \emptyset$     %The set of solution
- While  $A$  does not form a spanning tree
  - find a cut  $(X, V - X)$   
*with no edges from  $A$  crossing the cut*
  - $e$ : a *smallest* edge crossing the cut
  - $A \leftarrow A \cup \{e\}$

## Kruskal's algorithm implements GenericMST

- $e = (u, v)$ :
- The cut:  $X =$  vertices connected to  $u$  by edges in  $A$ .
- $v \notin X$  (so  $v \in V - X$ )
- No edge in  $A$  crossing  $(X, V - X)$
- $e$  has smallest weight amongst all edges crossing  $(X, V - X)$ .

## Correctness of GenericMST

$A_i$ : partial solution obtained after  $i$  iterations.

**Promising partial solution:**  $A_i$  is promising if for some minimum spanning tree  $T$  extends  $A_i$ :

$$A_i \subseteq T$$

Show that  $A_i$  is promising for all  $i \leq n$ .

Then  $A_n$  must be an optimal solution.

## Induction step

Assume that  $A_i$  is promising, i.e.,

$$A_i \subseteq T$$

for some minimum spanning tree  $T$ .

- $A_{i+1} = A_i$ : Then  $A_{i+1}$  is also promising – DONE !
- $A_{i+1} = A_i \cup \{e\}$ 
  - $e \in T$ : then  $A_{i+1} \subseteq T$  DONE !
  - $e \notin T$

- Then  $T \cup \{e\}$  contains a cycle  $C$
- $C$  must cross the cut  $(X, V - X)$  by another edge  $e'$
- $w(e) \leq w(e')$
- $T' = (T - \{e\}) \cup \{e'\}$

## Longest Common Subsequence

**Input:** Two sequences (arrays)  $X[1 \dots n]$ ,  $Y[1 \dots m]$ .

**Output:** A longest common subsequence of  $X$ ,  $Y$ .

$$S_{i,j} = LCS(X[1 \dots i], Y[1 \dots j])$$

**Observation:**

$$S_{n,m} = \begin{cases} S_{n-1,m-1} \circ X[n] & \text{if } X[n] = Y[m] \\ \text{longer of } \{S_{n-1,m}, S_{n,m-1}\} & \text{otherwise} \end{cases}$$

## A Recursive Program

LCS( $X, Y, n, m$ ):

1. If  $n = 0$  or  $m = 0$ : return  $\emptyset$
2. If  $X[n] = Y[m]$  return  $LCS(X, Y, n - 1, m - 1) \circ X[n]$
3. Else
4.      $L_1 = LCS(X, Y, n - 1, m)$
5.      $L_2 = LCS(X, Y, n, m - 1)$
6.     If  $|L_1| > |L_2|$
7.         return  $L_1$
8.     Else
9.         return  $L_2$
10.     End If
11. End If

**Running Time** (Assume  $n \leq m$ ):

$$\Omega(2^n)$$

**Reason:** Repeated calls to small (sub)problems

$$LCS(X, Y, i, j) \quad \text{for } 1 \leq i \leq n, 1 \leq j \leq m$$

There are only  $n \times m$  subproblems.

**Improvement:** Solve the subproblems *once* and *store* their solutions !

LCS2( $X, Y, n, m$ ):

1.  $S$ :  $n \times m \times n$  array % For storing solution.
2. For  $j = 1$  to  $m$  do  $S[0, j] \leftarrow \emptyset$
3. For  $i = 1$  to  $n$  do  $S[i, 0] \leftarrow \emptyset$
4. For  $i = 1$  to  $n$  do
5.     For  $j = 1$  to  $m$  do
6.         If  $X[i] = Y[j]$  do  $S[i, j] \leftarrow S[i - 1, j - 1] \circ X[i]$
7.         Else  $S[i, j] \leftarrow$  longer of  $\{S[i - 1, j], S[i, j - 1]\}$
8.         End If
9.     End For
10. End For
11. Return  $S[n, m]$

**Space:**  $n \times m \times n$

**Observation:** No need to store  $S[i, j]$

Can just store the *lengths* of  $S[i, j]$ , and *recover*  $S[n, m]$  at the end.

Computing the length  $L_{i,j}$  of  $S_{i,j}$ :

$$L_{0,j} = L_{i,0} = 0 \quad \text{for } 0 \leq i \leq n, 0 \leq j \leq m \quad (1)$$

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & \text{if } X[n] = Y[m] \\ \max\{L_{i-1,j}, L_{i,j-1}\} & \text{otherwise} \end{cases} \quad (2)$$

## Program to compute $L$ :

1.  $L$ :  $n \times m$  array
2. For  $j = 1$  to  $m$  do  $L[0, j] \leftarrow 0$
3. For  $i = 1$  to  $n$  do  $L[i, 0] \leftarrow 0$
4. For  $i = 1$  to  $n$  do
5.     For  $j = 1$  to  $m$  do
6.         If  $X[i] = Y[j]$ :  $L[i, j] \leftarrow L[i - 1, j - 1] + 1$
7.         Else  $L[i, j] \leftarrow \max \{L[i - 1, j], L[i, j - 1]\}$
8.         End If
9.     End For
10. End For

**Compute a longest common subsequence using  $L$ :**

**Idea:** examine how  $L[i, j]$  are computed in order to decide whether to include  $X[i]$  in the common sequence.

Print-LCS( $X, Y, L, n, m$ ):

1.  $\ell \leftarrow L[n, m]$     % length of a longest common subsequence of  $X$  and  $Y$ .
2.  $Z$ : array of length  $\ell$     % the output common subsequence

3.  $i \leftarrow n, j \leftarrow m$
4. While  $l > 0$  do
5.     If  $X[i] = Y[j]$  then
6.          $Z[l] \leftarrow X[i]$
7.          $i \leftarrow i - 1, j \leftarrow j - 1, l \leftarrow l - 1$
8.     Elseif  $L[i, j] = L[i - 1, j]$  then
9.          $i \leftarrow i - 1$
10.     Else  $j \leftarrow j - 1$
11.     End If
12. End While
13. Output  $Z$ .