Advanced Computer Contest Preparation

Lecture 16

STRING DP

Interleaving Strings

- We are given 3 strings, A, B, and C
- Is C an interleaving of A and B?
- A string interleaving of A and B is formed by taking characters of B and inserting them into A, preserving their original order in B
- Examples:
  - A = "ABC", B = "XYZ", C = "AXBYCZ"
  - A = "AAY", B = "AAZ", C = "AAAYAZ"
- Assume $|C| = |A|+|B|$
  - $|S|$ denotes the length of string S

DP Solution

- The overall problem is:
  - Is C an interleaving of A and B?
- The subproblems are:
  - Is a prefix of C an interleaving of a prefix of A of length $i, for\ i = 0\ to\ |A|$, and a prefix of B of length $j, for\ j = 0\ to\ |B|$?
  - A *prefix* of a string is a substring that starts at the first character

DP Solution

- Let $p(i,j)$ represent whether or not a prefix of C of length $i+j$ is an interleaving of a prefix of A of length i and a prefix of B of length j
  - $p(i,j)$ is either true or false
- If $A_i = C_{i+j}$ and $p(i-1,j)$ is true, $p(i,j)$ is true
- If $B_j = C_{i+j}$ and $p(i,j-1)$ is true, $p(i,j)$ is true
- Otherwise, $p(i,j)$ is false
- Base case: $p(0,0)$ = true

Example 1

- A = “AAY”, B = “AAZ”, C = “AAAYAZ”

$i(A) \setminus j(B)$	0	1	2	3
0	1	1	1	0
1	1	1	1	0
2	1	1	0	0
3	0	1	1	1

Example 2

- A = “AAY”, B = “AAZ”, C = “AAYAZA”

$i(A) \setminus j(B)$	0	1	2	3
0	1	1	1	0
1	1	1	0	0
2	1	0	0	0
3	1	1	0	0

Pseudocode

```
string A,B,C;
//get input, assume |C| = |A|+|B|
bool dp[][];
dp[0][0] = 1;
for (int i = 0; i <= A.length(); i++){
       for (int j = 0; j <= B.length(); j++){
              if (i > 0 && A[i-1] == C[i+j-1] && dp[i-1][j])
                     dp[i][j] = 1;
              if (j > 0 && B[j-1] == C[i+j-1] && dp[i][j-1])
                     dp[i][j] = 1;
       }
}
print dp[A.length()][B.length()];
```

Analysis

- How many states are there?
  - $O(|A||B|)$
- How many subproblems does each state depend on?
  - $O(1)$
- What is the time complexity needed to compute the solution to a problem?
  - $O(1)$
- Therefore, the final time complexity is $O(|A||B|)$

Longest Common Subsequence

- We are given two strings, X and Y
- What is the length of the longest common subsequence (LCS) between X and Y?
- Recall: to form a *subsequence* of an original sequence, we delete some elements without changing order
- A common subsequence between X and Y is any subsequence present in both X and Y
- Example:
  - X = "ABCBDA"
  - Y = "BDCABA"
  - LCS = "BCBA"

DP Solution

- The overall problem is:
  - What is the length of the LCS between strings X and Y?
- The subproblems are:
  - What is the length of the LCS between a prefix of X of length $i, for\ i = 0\ to\ |X|$, and a prefix of Y of length $j, for\ j = 0\ to\ |Y|$?

DP Solution

- Let $p(i,j)$ represent the length of the LCS between a prefix of X of length i and a prefix of Y of length j
- Base case: $p(i,j) = 0$ if $i = 0$ or $j = 0$
  - There cannot be an LCS if one of the strings is empty

DP Solution

- If $X_i = Y_j$
  - $p(i,j) = p(i-1,j-1)+1$
  - Extend an existing LCS by 1
- Else
  - $p(i,j) = max(p(i-1,j),p(i,j-1))$
  - Choose the longest existing LCS

Example

	j	0	1	2	3	4	5	6
i			B	D	C	A	B	A
0		0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4

Pseudocode

```
string X,Y;
int dp[][];
//get input;
for (int i = 0; i <= X.length(); i++){
       for (int j = 0; j <= Y.length(); j++){
              if (i == 0 || j == 0) dp[i][j] = 0;
              else if (X[i-1] == Y[j-1])
                     dp[i][j] = dp[i-1][j-1]+1;
              else
                     dp[i][j] = max(dp[i-1][j],dp[i][j-1]);
       }
}
print dp[X.length()][Y.length()]
```

Analysis

- How many states are there?
  - $O(|X||Y|)$
- How many subproblems does each state depend on?
  - $O(1)$
- What is the time complexity needed to compute the solution to a problem?
  - $O(1)$
- Therefore, the final time complexity is $O(|X||Y|)$

Edit Distance

- We are given two strings X and Y
- We want to change X into Y using the following operations:
  - Insert: insert a character anywhere into X
  - Delete: remove a character anywhere from X
  - Substitute: change a character in X
- What is the minimum number of operations required?

DP Solution

- The overall problem is:
  - What is the minimum number of operations required to change X into Y?
- The subproblems are:
  - What is the minimum number of operations required to change a prefix of X of length i, $for\ i = 0\ to\ |X|$, into a prefix of Y of length j, $for\ j = 0\ to\ |Y|$?

DP Solution

- Let $p(i,j)$ be the minimum cost required to change a prefix of X of length i into a prefix of Y of length j
- Case 1: $i = 0$
  - $p(i,j) = j$
  - We need to perform j insertions from an empty string
- Case 2: $j = 0$
  - $p(i,j) = i$
  - We need to perform i deletions to form an empty string

DP Solution

- Case 3: $X_i = Y_j$
  - $p(i,j) = p(i-1,j-1)$
    - No operation required
- Case 4: $X_i \neq Y_j$
  - $p(i,j) = min($
  - $p(i-1,j)+1$
    - Delete the i^{th} character of X
  - $p(i,j-1)+1$
    - Insert the j^{th} character of Y
  - $p(i-1,j-1)+1$
    - Substitute the i^{th} character of X for the j^{th} character of Y

  $)$

Example

	j	0	1	2	3	4	5	6	7	8
i			T	H	U	R	S	D	A	Y
0		0	1	2	3	4	5	6	7	8
1	T	1	0	1	2	3	4	5	6	7
2	U	2	1	1	1	2	3	4	5	6
3	E	3	2	2	2	2	3	4	5	6
4	S	4	3	3	3	3	2	3	4	5
5	D	5	4	4	4	4	3	2	3	4
6	A	6	5	5	5	5	4	3	2	3
7	Y	7	6	6	6	6	5	4	3	2

Pseudocode

```
string X,Y;
int dp[][];
//get input
for (int i = 0; i <= X.length(); i++){
        for (int j = 0; j <= Y.length(); j++){
                if (i == 0) dp[i][j] = j;
                else if (j == 0) dp[i][j] = i;
                else if (X[i-1] == Y[j-1]) dp[i][j] = dp[i-1][j-1];
                else{
                        dp[i][j] = min( dp[i-1][j]+1,
                                dp[i][j-1]+1, dp[i-1][j-1]+1);
                }
        }
}
print dp[X.length()][Y.length()]
```

Analysis

- How many states are there?
  - $O(|X||Y|)$
- How many subproblems does each state depend on?
  - $O(1)$
- What is the time complexity needed to compute the solution to a problem?
  - $O(1)$
- Therefore, the final time complexity is $O(|X||Y|)$

THANK YOU!