CMPSC 250

Analysis of algorithms

Dr. Aravind Mohan

Allegheny College
Dynamic Programming

What is Dynamic Programming?

Dynamic Programming - Solving a problem by breaking it down divide-and-conquer style into a set of smaller programs. Only solve each subproblem once, no repeat computations.
Dynamic Programming

Why Dynamic Programming?
Solving certain algorithmic problems take exponential time with a recursive solution. Leveraging divide and conquer principle, finding optimal solutions to sub problems through dynamic programming may leads us to $O(n^2)$ or a better solution than brute force approach.
Approach

- Design a top-down recursive algorithm.
- Convert it to a bottom-up iterative algorithm
Toy Example: Fibonacci

The Fibonacci Sequence

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377…

1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8
5 + 8 = 13
8 + 13 = 21
13 + 21 = 34
21 + 34 = 55
34 + 55 = 89
55 + 89 = 144
89 + 144 = 233
144 + 233 = 377
Toy Example: Recursive

```c
int fibo(int val) {
    if (val == 0 || val == 1) {
        return 1;
    } else {
        return fibo(val-1) + fibo(val-2);
    } //if-else
} //fibo
```
int fibo(int stop) {
    int val[0] = 0, val[1] = 1;
    for (int i = 2; i <= stop; i++) {
        val[i] = val[i-1] + val[i-2];
    } //for
    return val[stop];
} //fibo
Subset Sum Problem

- Input coin values: $v_1, v_2, \ldots, v_n$
- Total value $L$
- Output: Change for $L$ cents, if possible

Example Input: 7, 2, 11, 9, 16, 13;  = 41
Example Output: 11, 16, 14 or 2, 9, 16, 14
Start with a recursive solution

```java
isSubsetSum(int set[], int n, int sum) {
    if (sum == 0) {
        return true;
    } //if
    if (n == 0 && sum != 0) {
        return false;
    } //if
    if (set[n-1] > sum) {
        return isSubsetSum(set, n-1, sum);
    } //if
    return isSubsetSum(set, n-1, sum) ||
           isSubsetSum(set, n-1, sum-set[n-1]);
} //isSubsetSum
```
Subset Sum Problem

Convert to an iterative solution

```c
isSubsetSum(int set[], int n, int sum) {
    for (int i=0; i<sum; i++) {
        subsetsum[0][i] = 0;
    } //for
    for (int i=0; i<n; i++) {
        subsetsum[i][0] = 1;
    } //for
    for (int i=1; i<n; i++) {
        for (int j=1; j<sum; i++) {
            subsetsum[i][j] = max(
                subsetsum[i-1][j],
                subsetsum[i-1][j-set[j]]
            );
        } //for
    } //for
    return subsetsum[n][sum];
} //isSubsetSum
```
# Subset Sum Problem

## Example

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1
Subset Sum Problem

Example

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Longest Common Subsequence Problem

Input:
- String $A = A_1 A_2 \cdots A_m$
- String $B = B_1 B_2 \cdots B_n$

Output:
- Length of the longest common subsequence (can skip letters).
- Example input: $A = xxyzzxyz$, $B = xyxxxyz$
- Example output: 5 (xyxyz)
Longest Common Subsequence Problem

Start with a recursive solution

```cpp
LCS(int m, int n) {
    if (m == 0 || n == 0) {
        return 0;
    } else if (A[m] == B[n]) {
        return LCS(m-1, n-1) + 1;
    } else {
        return max(LCS(m, n-1), LCS(m-1, n));
    } //if-else
} //LCS
```
Longest Common Subsequence Problem

Convert to a iterative solution

```cpp
LCS(int m, int n) {
    for (int i = 0; i <= m; i++)
        LCS[i][0] = 0;
    for (int i = 0; i <= n; i++)
        LCS[0][i] = 0;
    for (int i = 1; i <= m; i++)
        for (int j = 1; j <= n; j++)
            if (A[i] == B[j])
                LCS[i][j] = LCS[i-1][j-1]+1;
            else
                LCS[i][j] = max(LCS[i][j-1], LCS[i-1][j]);
    return LCS[m][n];
} //LCS
```
### Longest Common Subsequence Problem

#### Example

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>x</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Longest Common Subsequence Problem

Example Recover String

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The arrows indicate the path to recover the strings.
Read article on dynamic programming on course webpage.
Class Activity - Post on slack

- Find longest subsequence ABCDEFGH ACXFBCDYGTH
- Find subset sum on sequence 4, 1, 5, 10, 11, 2, 3 with Sum = 8