Day 17 Notes

Zach Neveu

June 4, 2019

1 Agenda

- Quiz
- Review of Dynamic Programming

2 Dynamic Programming

- Problem: order of matrix multiplication
- Key concepts
  - All solutions must have a final multiplication
  - Final multiplication cost is LHS+RHS+combining
  - For optimal soln, LHS, RHS must each be optimal
  - This is true recursively
- Notation
  - $m[i,j]$ - optimal sub-product cost $\prod_{k=i}^{j} A_k$
  - $m[i,j] = 0$ if $i == j$, else $argmin_k(m[i,k] + m[k+1,j] + p_{i-1}p_k p_j)$

```
# Recursive Matrix Chain
# Computes m[i,j]
def RMC(p,i,j):
    if i == j:
        return 0
    else:
        m[i,j] = MAXINT
        for k in range(i,j-1):
            q = RMC(p,i,k)+RMC(p,k+1,j)+p[i-1]*p[k]*p[j]
            if q < m[i,j]:
                m[i,j] = q
    return m[i,j]
```

- Quiz question: modify this algorithm to keep track of best k at each step!
- Problem: Same sub-problems are solved over-and-over
- Just like recursive Fibonacci

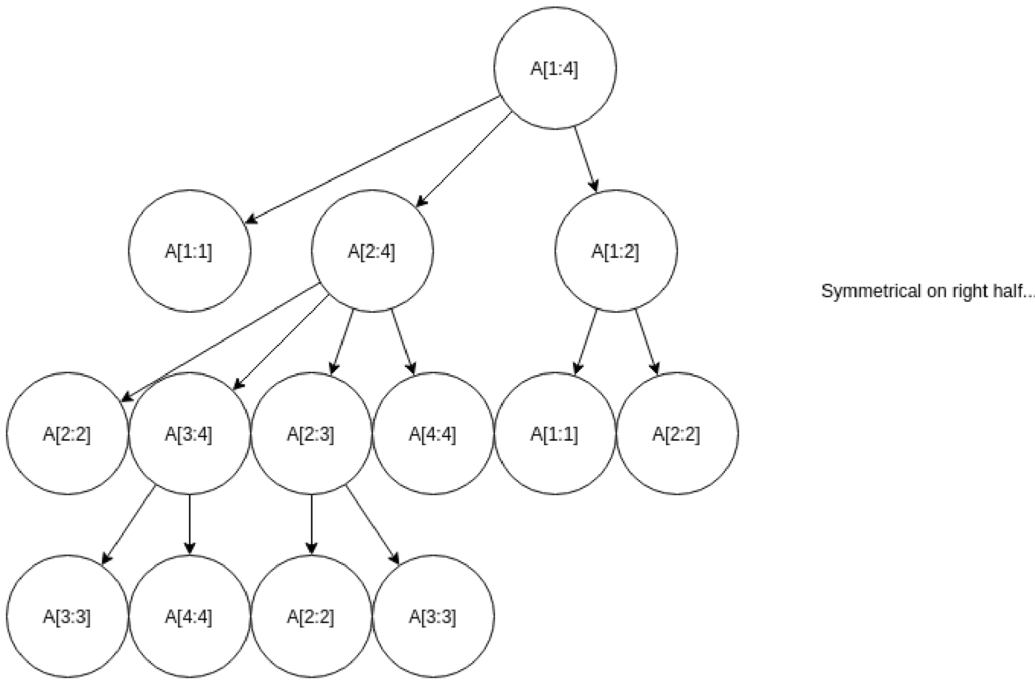

Figure 1: Example RMC solution tree for $A_{1..4}$

- Solution: make $m[]$ global, and check if soln to subproblem has already been found. Use answer if it has.
- **memoization**: This process of saving already-computed solutions

Complexity

- m matrix has n^2 entries
- Each entry computed exactly once
- Work to compute each entry (or at least one half (diagonal symmetry)):
  - Each func call takes $O(n)$
  - Total function takes $O(n * n^2) = O(n^3)$

Optimization

- Idea: instead of discovering tree recursively, why not just go through matrix in order and solve each subproblem?
- Organize as triangle (half matrix) w/ i,j along edges
- Build values from trivial diagonal to corner
- **dynamic programming** technically refers to this triangle approach, not the memoization

Dynamic TSP Example

- Is set of consecutive indices in HC also a HC? NO!
- Instead, given a subset of nodes, s, and some $k \in s$, $c(s,k)$ = min cost to start at node 1, visit all nodes in S, and end at k.
- $c()$ can be expressed in terms of subproblems easily.
- Remove k^{th} node from the set and repeat with new k
- Try all k and pick optimal
- Example
  - c({2,4,6,7}, 2) is min(c({4,6,7},4)+w[4,2], c({4,6,7}, 6)+w[6,2], ...)
  - In general, $c(S,k) = min_{m \in S-\{k\}} c(s-\{k\},m) + w[m,k]$