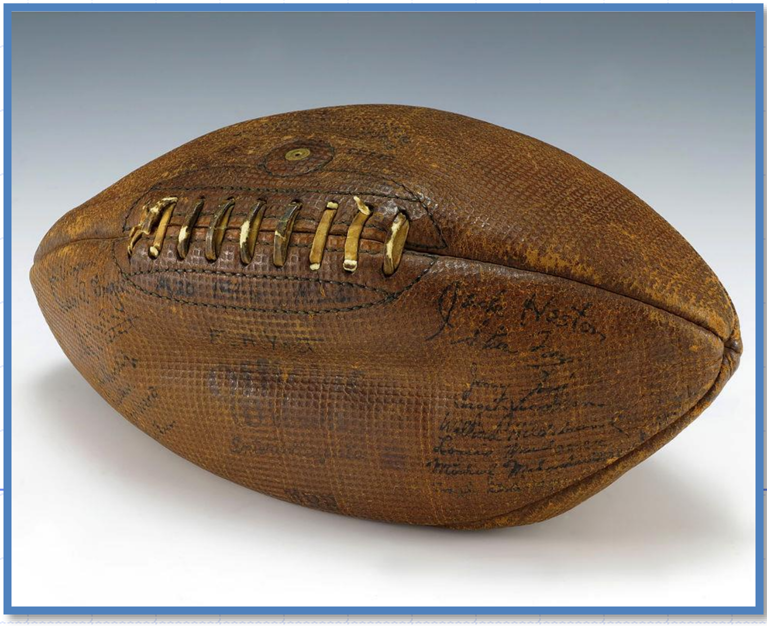
Presentation for use with the textbook, Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia, Wiley, 2015

Dynamic Programming: Game Strategies



Football signed by President Gerald Ford when playing for University of Michigan. Public domain image.

Coins in a Line

- "Coins in a Line" is a game whose strategy is sometimes asked about during job interviews.
- In this game, an even number, n, of coins, of various denominations, are placed in a line.
- Two players, who we will call Alice and Bob, take turns removing one of the coins from either end of the remaining line of coins.
- The player who removes a set of coins with larger total value than the other player wins and gets to keep the money. The loser gets nothing.
- Alice's goal: get the most.

Alice: (1, $6), (6, $5), (4, $7)   Total value = $18

Bob: (2, $5), (5, $3), (3, $2)   Total value = $10

Figure 12.7: The coins-in-a-line game. In this instance, Alice goes first and ultimately ends up with $18 worth of coins. U.S. government images. Credit: U.S. Mint.

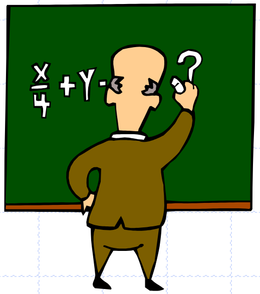
False Start 1: Greedy Method

- A natural **greedy** strategy is "always choose the largest-valued available coin."
- But this doesn't always work:
  - [5, 10, 25, 10]: Alice chooses 10
  - [5, 10, 25]: Bob chooses 25
  - [5, 10]: Alice chooses 10
  - [5]: Bob chooses 5
- Alice's total value: 20, Bob's total value: 30. (Bob wins, Alice loses)

False Start 2: Greedy Method

- Another **greedy** strategy is "choose odds or evens, whichever is better."
- Alice can always win with this strategy, but won't necessarily get the most money.
- Example: [1, 3, 6, 3, 1, 3]
- Alice's total value: $9, Bob's total value: $8.
- Alice wins $9, but could have won $10.
- How?

The General Dynamic Programming Technique



- Applies to a problem that at first seems to require a lot of time (possibly exponential), provided we have:
  - **Simple subproblems:** the subproblems can be defined in terms of a few variables, such as j, k, l, m, and so on.
  - **Subproblem optimality:** the global optimum value can be defined in terms of optimal subproblems
  - **Subproblem overlap:** the subproblems are not independent, but instead they overlap (hence, should be constructed bottom-up).

Defining Simple Subproblems

- Since Alice and Bob can remove coins from either end of the line, an appropriate way to define subproblems is in terms of a range of indices for the coins, assuming they are initially numbered from 1 to n.
- Thus, let us define the following indexed parameter:

$$M_{i,j} = \begin{cases} \text{the maximum value of coins taken by Alice, for coins} \\ \text{numbered } i \text{ to } j \text{, assuming Bob plays optimally.} \end{cases}$$

Therefore, the optimal value for Alice is determined by $M_{1,n}$.

Subproblem Optimality

- Let us assume that the values of the coins are stored in an array, V, so that coin 1 is of Value V[1], coin 2 is of Value V[2], and so on.
- Note that, given the line of coins from coin i to coin j, the choice for Alice at this point is either to take coin i or coin j and thereby gain a coin of value V[i] or V[j].
- Once that choice is made, play turns to Bob, who we are assuming is playing optimally.
  - We should assume that Bob will make the choice among his possibilities that minimizes the total amount that Alice can get from the coins that remain.

Subproblem Overlap

- Alice should choose based on the following:

  - If $j = i + 1$, then she should pick the larger of $V[i]$ and $V[j]$, and the game is over.
  - Otherwise, if Alice chooses coin i, then she gets a total value of
  $$\min\{M_{i+1,j-1},\ M_{i+2,j}\} + V[i].$$
  - Otherwise, if Alice chooses coin j, then she gets a total value of
  $$\min\{M_{i,j-2},\ M_{i+1,j-1}\} + V[j].$$

- That is, we have initial conditions, for i=1,2,…,n-1:

$$M_{i,i+1} = \max\{V[i], V[i+1]\}.$$

- And general equation:

$$M_{i,j} = \max\left\{\min\{M_{i+1,j-1},\ M_{i+2,j}\} + V[i],\ \min\{M_{i,j-2},\ M_{i+1,j-1}\} + V[j]\right\}.$$

Analysis of the Algorithm

- We can compute the $\mathbf{M}_{i,j}$ values, then, using memoization, by starting with the definitions for the above initial conditions and then computing all the $\mathbf{M}_{i,j}$'s where $j - i + 1$ is 4, then for all such values where $j - i + 1$ is 6, and so on.
- Since there are $\mathbf{O}(\mathbf{n})$ iterations in this algorithm and each iteration runs in $\mathbf{O}(\mathbf{n})$ time, the total time for this algorithm is $\mathbf{O}(\mathbf{n}^2)$.
- To recover the actual game strategy for Alice (and Bob), we simply need to note for each $\mathbf{M}_{i,j}$ whether Alice should choose coin i or coin j.

Algorithm for Coins-in-a-Line from Lecture

Algo: Coins (C)
Input: list C of integer n≥ 0 coins
Output: maximum value that Alice, the first player, can obtain from C
assuming both players play perfectly

```
M ← new n x n Matrix
for i← to n
   M[i, i] ← C(i)   // only one coin, pick it (base case)

for i ← 1 to (n-1) do // two coins to choose from (base case)
       M[i, i+1] ← max { C(i), C(i+1) }

for Length ← 2 to (n-1) do
   for i ← 1 to (n – Length)
      j ← i + Length
      M[i, j] ← max {
                    C(i) + { min { M[i+2, j], m[i+1, j-1] } ,
                    C(j) + { min { M[i+1, j-1], M[i, j-2] } }
return M[1,n]
```