Recitation 18

Treasureship!

The new boardgame Treasureship is played by placing 2×1 ships within a $2 \times n$ rectangular grid. Just as in regular battleship, each 2×1 ship can be placed either horizontally or vertically, occupying exactly 2 grid squares, and each grid square may only be occupied by a single ship. Each grid square has a positive or negative integer value, representing how much treasure may be acquired or lost at that square. You may place as many ships on the board as you like, with the score of a placement of ships being the value sum of all grid squares covered by ships. Design an efficient dynamic-programming algorithm to determine a placement of ships that will maximize your total score.

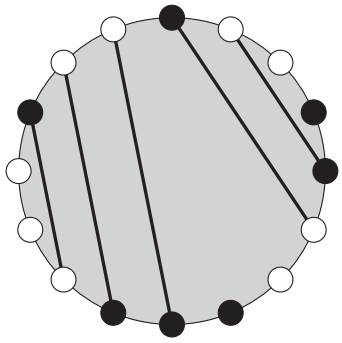
Solution:

1. **Subproblems**
   - Game board has n columns, each of height 2, and 2 rows, each of width n.
   - Let $v(x, y)$ denote the grid value at row y column x, for $0 \leq y \leq 1$, $0 \leq x \leq n$
   - Let $s(i, j)$ represent game board subset containing columns 1 to i of row 1, and columns 1 to $i + j$ of row 2, for $-1 \leq j \leq 1$.
   - $\boxed{x(i, j)\text{: maximum score, only placing ships on board subset } s(i, j)}$
2. **Relate**
   - Either you get points from every grid square in column i or not
   - $$x(i,j) = \begin{cases} \max\{v(i,1) + v(i-1,1) + x(i-2,+1), x(i-1,0)\} & \text{if } j = -1 \\ \max\{v(i+1,2) + v(i,2) + x(i,-1), x(i,0)\} & \text{if } j = 1 \\ \max\{v(i,1) + v(i,2) + x(i-1,0), x(i,-1), x(i-1,+1)\} & \text{if } j = 0 \end{cases}$$
   - Subproblems $x(i, j)$ only depend on strictly lexiographically smaller (i, j), so acyclic.
3. **Base**
   - $s(i, j)$ contains $2i + j$ grid squares
   - $x(i, j) = 0$ if $2i + j < 2$ (can't place a ship if fewer than 2 squares!)
4. **Solution**
   - Solution is $x(n, 0)$, the maximum considering all grid squares.
   - Store parent pointers to reconstruct ship locations

5. **Time**
    - # subproblems: $O(n)$
    - work per subproblem $O(1)$
    - $O(n)$ running time

Wafer Power

A start-up is working on a new electronic circuit design for highly-parallel computing. Evenly-spaced along the perimeter of a circular wafer sits n ports for either a power source or a computing unit. Each computing unit needs energy from a power source, transferred between ports via a wire etched into the top surface of the wafer. However, if a computing unit is connected to a power source that is too close, the power can overload and destroy the circuit. Further, no two etched wires may cross each other. The circuit designer needs an automated way to evaluate the effectiveness of different designs, and has asked you for help. Given an arrangement of power sources and computing units plugged into the n ports, describe an $O(n^3)$ time dynamic programming algorithm to match computing units to power sources by etching non-crossing wires between them onto the surface of the wafer, in order to maximize the number of powered computing units, where wires may not connect two adjacent ports along the perimeter. Below is an example wafer, with non-crossing wires connecting computing units (white) to power sources (black).



Solution:

1. **Subproblems**
    - Let a_i be the type of object in port p_i around the circle, for $i \in \{1, \ldots, n\}$.
    - Want to match opposite ports which can be connected by non-intersecting wires.
    - $x(i, j)$: maximum number of matchings when restricted to ports in the interval from p_i to p_j, for i, j such that $1 \leq i \leq j \leq n$.
2. **Relate**

- Guess what port to match with first port in interval, either:
  - first port does not match, try to match the rest;
  - first port matches with last port, try to match the rest; or
  - first port matches with a port in the middle, try to match in either side.
- Non-adjacency condition restricts possible matchings with i to $t \in \{i+2, \ldots, j\}$, except when $(i, j) = (1, n)$ where $t \in \{3, \ldots, n-1\}$, as 1 cannot match with n.
- Let $m(i, j) = 1$ if $a_i \neq a_j$ and $m(i, j) = 0$ otherwise
- $x(i, j) = \max \left\{ \begin{array}{l} x(i+1, j), \\ m(i, j) + x(i+1, j-1), \\ \max_{t \in \{i+2, \ldots, j-1\}} (m(i, t) + x(i+1, t-1) + x(t+1, j))) \end{array} \right\}$
- for $i \in \{1, \ldots, n\}$ and $j \in \{i, \ldots, n\}$, except for $(i, j) = (1, n)$ where:
- $x(1, n) = \max\{x(2, n), \max_{t \in \{3, \ldots, n-1\}} (m(1, t) + x(2, t-1) + x(t+1, n))\}$
- Subproblems $x(i, j)$ only depend on strictly smaller $j - i$, so acyclic
- Solve in order of increasing $j - i \in \{0, \ldots, n-1\}$, then in increasing $i \in \{1, \ldots, n\}$

3. **Base Cases**
   - $x(i, j) = 0$ for $j - i = 0$ (one port can't match!)
4. **Solution**
   - Solve subproblems via recursive top down or iterative bottom up.
   - Solution to original problem is $x(1, n)$.
   - Can store t if matching at t achieved maximum or `None` if no match acheived maximum, in order to reconstruct matching.
5. **Time**
   - # subproblems: $O(n^2)$
   - work per subproblem: $O(n)$
   - $O(n^3)$ running time