Sketch of Solutions for UMD HSPC 2016

(By: David Mount, Evan Golub, Amol Deshpande)

A A

3. Forced Cards

Problem:

- Given an array of (possibly negative) numbers deck[n], find the subarray deck[start..end] that has the maximum sum

Solution:

- There are many solutions. This one is quick, but not very efficient.

```
start = 0; end = -1; maxSum = 0;          // initialize default solution
for (i = 0; i < n; i++) {                 // try all possible starts
        sum = 0;
        for (j = start; j < n; j++) {     // try all possible ends
                sum += deck[j]            // update sum
                if (sum > maxSum) { start = i; end = j; maxSum = sum; }
        }
}
```

- This takes $O(n^2)$ time.

3. Forced Cards

Problem:

- Given an array of (possibly negative) numbers deck[n], find the subarray deck[start..end] that has the maximum sum

Solution:

- The approach presented on the previous page takes $O(n^2)$ time.
- The problem can be solved in $O(n)$ time. To approach solving this problem in linear time, consider that as you pass through the information you can make some local decisions.
  - For inspiration, consider minimum finding. You can call the first element in the smallest so far and store it. Then, as you traverse the rest of the list, if you see something smaller, then it is the smallest so far so you should store that instead. Due to transitivity, you never have to "look back" with this. At the end, the smallest so far is the true smallest.
  - Something useful to think about for this problem includes the fact that if you have a running sum from a starting point, if it ever goes negative then you've passed the best ending point for that starting point.

4. Legal Moves in Go

Problem:

- Given a board configuration in Go, is a proposed move legal?

Solution:

- Each square is labeled as White, Black, or Empty.
- Suppose that the move is White (since Black is symmetrical)
- If any of the four neighbors is empty, this move is legal
- Otherwise, set this square to White, and start a recursive function to check for liberties:
  - If any neighboring square is empty, return true
  - If not, mark this square as visited, and recurse on all unvisited neighbors
  - If for any unvisited white neighboring stone, the recursive call returns a "true", then return "true"
  - If for any unvisited black neighboring stone, the recursive call returns a "false", then return "true"
  - If neither of the above is true, return "false"
- Some more bookkeeping needs to be done to return why the move is legal

5. Scavenger One

Problem:

- Given a list of the weights of 30 to 40 items and a cargo capacity, what is the combination of items that gets you that exact weight?

Solution:

- The most straight-forward approach is to create a recursive helper method and essentially at each point through the recursion you'll ask two questions if you aren't down to having just one item to consider:
  - Can I get that weight using the first item in the list I have and then ask the question on the remainder of the list and the capacity minus the weight of this first item.
  - Can I get that weight without using the first item on the list I have and then ask the question on the remainder of the list and the current value for the capacity limit.
- Yes, this has an exponential explosion of calls possible but that's why the number of items was limited. In fact, a polynomial solution for this type of problem is currently an open question.

6. Tree Ragnorism

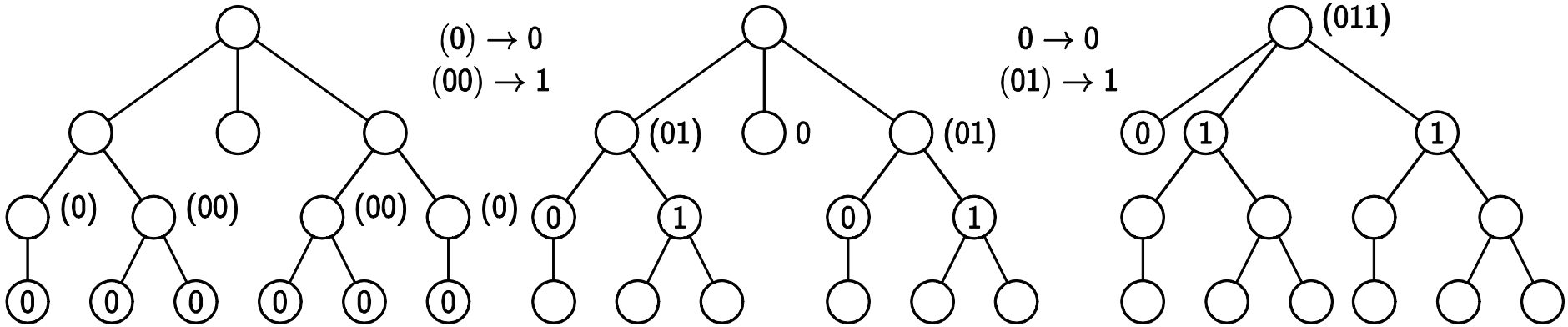
Problem:

- Given two trees, are they isomorphic?

Solution:

- Fix the first tree, and try all choices of root node for the second tree. Then apply the procedure below for rooted tree isomorphism.
- Label nodes by their level, where the root is at level 0. If both trees have different numbers of levels, then fail.
- Starting at the lowest level, label all nodes with the same label (0).
- Apply the following process recursively:
  - For each node, sort its children by label number and form a sequence
  - Sort these sequences, and assign all sequences with the same value the same (new) label value
- If the roots have the same label, then the trees are isomorphic

6. Tree Ragnorism



7. Combination Puzzle

Problem:

- Set a number of dials to 0, where turning one dial automatically turns the next dial in the opposite direction

Solution:

- Let dial[i] be the initial dial setting (0..9)
- Let x[i] denote the number of increments of dial i (mod 10)
- Incrementing dial i-1 causes dial i to decrement:
  - (x[i-1] - x[i] = dial[i]) mod 10 implies (x[i] = x[i-1] – dial[i]) mod 10
- Suppose that we know x[0]. We infer the remaining dial increments:
  - x[1] = (x[0] - dial[1]) mod 10
  - x[2] = (x[1] – dial[2]) mod 10
- We will guess all possible choices for x[0] and work the others out
- The number of moves for dial i is min(x[i], 10 – x[i])
- Take the choice for x[0] that leads to the lowest cost

8. Opening Doors

Problem:

- You have a set of m switches, each toggles a subset of n doors. You want to set the toggles to open a given set of doors

Solution:

- Suppose there are 5 doors. Write each switch as a binary vector:

0	1	2	3	4	Doors
1	1	0	1	1	{0, 1, 3, 4} – Switch 1
0	1	1	0	1	{1, 2, 4} – Switch 2
0	1	0	1	0	{1, 3} – Switch 3
…					

- Let A be an $m \times n$ matrix, where $A_{i,j} = 1$ if switch i toggles door j
- Let x be a binary m-vector, where $x_i = 1$ if we select switch i
- Let b be a binary n-vector, where $b_j = 1$ if we want door j opened
- We seek a 0-1 vector x such that $x \cdot A = b$ (using arithmetic mod 2)
- We can solve this using **Gaussian elimination** (over integers mod 2)

Thanks!

Questions?