CSCI-UA.0380-001
Programming Challenges

Sean McIntyre

Class 04: Greedy and Binary Search

Today's agenda

- Midterm next week
- Homework discussion
- Lecture
- Break (~3:00-3:15pm)
- Practice (~3:15pm-4:15pm)
- Discussion of problems

Homework discussion

Lotto

- Given a set S, print all combinations of the set
- E.g., S = { 1,2,3,5,8,13,21,34 }
  - 1,2,3,5,8,13

    1,2,3,5,8,21

    …

    3,5,8,13,21,34

Citizen Attention Offices

- Place 5 offices on the 5x5 city grid so that the total distance for people to go to the closest office is minimized
- Manhattan distance

Date Bugs

- There are N computers. Computer i resets its year counter to A_i whenever it reaches year B_i.
- Given the year displayed on each computer and all A_i *and* B_i what year is it now? (If possible.)
- Generalized Y2K bug

Ecosystem

- Find 3-member cyclic food chain
- Exercise for trying recursive graph traversal

Blocks

- Arrange *N* 1x1 cubes so that they form a rectangular block. What is the minimum surface area for the *N* cubes?

- Complete search

Greedy

Greedy algorithms

- Key idea: Make the choice that looks best at the moment
  - The hope: locally optimal choices lead to a globally optimal solution

- More: Wikipedia

Coin change problem

- With US coins (25, 10, 5, 1 cents), make change using the least possible number of coins.

Coin change problem

```
int total = 0;

while (x >= 25) {    // x is the number of cents to make change for
  total++;
  x -= 25;
}

while (x >= 10) {
  total++;
  x -= 10;
}

while (x >= 5) {
  total++;
  x -= 5;
}

while (x >= 1) {
  total++;
  x -= 1;
}
```

Coin change problem

```
// Recursive solution

int numberOfCoins(int x) {
  if (x >= 25)      return numberOfCoins(x-25)+1;
  else if (x >= 10) return numberOfCoins(x-10)+1;
  else if (x >= 5)  return numberOfCoins(x-5)+1;
  else if (x >= 1)  return numberOfCoins(x-1)+1;
  else              return 0;
}
```

Coin change problem

```
// Recursive solution

int numberOfCoins(int x) {
  if (x >= 25)      return numberOfCoins(x-25)+1;
  else if (x >= 10) return numberOfCoins(x-10)+1;
  else if (x >= 5)  return numberOfCoins(x-5)+1;
  else if (x >= 1)  return numberOfCoins(x-1)+1;
  else              return 0;
}

/*
    Stack trace:
    numberOfCoins(87)   // returns 6
    → numberOfCoins(62) // returns 5
    → numberOfCoins(37) // returns 4
    → numberOfCoins(12) // returns 3
    → numberOfCoins(2)  // returns 2
    → numberOfCoins(1)  // returns 1
    → numberOfCoins(0)  // returns 0
    Answer: 6
*/
```

Coin change problem

- Why this works:
  - It has optimal sub-structures
    - The optimal solution contains within it optimal solutions to its subproblems
  - It has a greedy property
    - The greedy choice + the optimal sub-structure produces the best solution (provable if given time)

Coin change problem

- It's possible to prove that the solution must contain the largest possible coin for each of the given values
  - numberOfCoins(87) = numberOfCoins(87 – 25) + 1
  - numberOfCoins(x) = numberOfCoins(x – largestCoin(x))+1

Coin change problem

- For many other coin denominations, the greedy solution does not work
- E.g., if we had a coin system with { 1, 3, 4 }, then the least number of coins to make 6 would be 3+3, not 4+1+1.

Coin change problem

- We'll revisit this problem later and discuss a smart complete search solution
  - Works regardless of the coin denominations
  - Uses dynamic programming

Traveling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?
  - Traveling Salesman Problem

Traveling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?
  - Traveling Salesman Problem
- Greedy choice: “At each stage visit an unvisited city nearest to the current city”

Traveling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?
  - Traveling Salesman Problem
- Greedy choice: “At each stage visit an unvisited city nearest to the current city”
  - Does not guarantee the best answer

Greedy choices in contest

- You have to be certain of your greedy choice
  - In contests, you might have to go off a hunch, try to find counterexamples
  - Generating a proof is not worth the effort
- Greedy solutions are typically compact and do not get TLEs
  - If you coded everything correctly, you should either expect “Accepted” or “Wrong Answer”

Greedy choices in contest

- If a complete search takes too long in a contest problem, it might be a sign that there is a greedy solution

Interval covering

- Given a list of intervals $[a_i, b_i]$, find the minimum number of intervals that covers the area from $[0, n]$ if such a covering exists

Interval covering

- Given a list of intervals $[a_i, b_i]$, find the minimum number of intervals that covers the area from $[0, n]$ if such a covering exists
- Greedy solution:
  - Choose the interval that covers 0 and extends furthest to the right
  - Continually choose an interval that overlaps with the previous interval and extends furthest to the right

Disjoint interval covering

- Given a list of intervals $[a_i, b_i]$, fit as many intervals in the area $[0, n]$ without any overlap between the intervals

Disjoint interval covering

- Given a list of intervals $[a_i, b_i]$, fit as many intervals in the area $[0, n]$ without any overlap between the intervals
- Greedy solution:
  - Choose the first interval with the earliest endtime
  - Choose the next interval that does not overlap with any previous interval with the earliest endtime

Greedy intervals

- For more on greedy and intervals, see here
  - I'll post the link on the website

Watering Grass problem

- A horizontal line of N sprinklers water a rectangle patch of grass of dimension W x L. Each sprinkler has an x_i-coordinate measured from the left of the rectangle patch and a radius r_i.
- What is the fewest number of sprinklers that can be used to water the entire rectangle?
- $1 <= N <= 10000$

Watering Grass problem

- Reduces down to interval covering

Binary search

Binary search

- Find a value in a sorted sequence
- E.g., { 0, 5, 13, 19, 22, 41, 55, 68, 72, 81, 98 }
  - Find the index of 55 (if it exists)

Binary search

```
binary_search(A, target):
    lo = 1, hi = size(A)
    while lo <= hi:
        mid = lo + (hi-lo)/2
        if A[mid] == target:
            return mid
        else if A[mid] < target:
            lo = mid+1
        else:
            hi = mid-1
    // if the code reaches here, target was not found
```

Binary search

- Binary search relies upon the sorted nature of the array list
- It relies upon the fact that we use a single function that returns false for all values LEFT of some index *A*, and true for all values RIGHT of *A*
  - A “monotonic function”

Binary search

- That means binary search doesn't have to be limited to arrays

Through the Desert

- You are an explorer trying to cross a desert with a jeep.
- There are events along the way:
  - Drive (consumes fuel)
  - New leak (wastes fuel)
  - Gas station (fill up gas tank)
  - Mechanic (fixes all leaks)

Through the Desert

- You want to find the smallest possible fuel tank that will take you across the desert, to 3 decimal points.
- The gas tank will be no larger than 10,000 units.

Through the Desert

- Complete search:
  - 10 million values, 0.000 to 10000.000

Through the Desert

- Complete search:
  - 10 million values, 0.000 to 10000.000
- Binary search:
  - You can create a function that will tell you whether or not a certain gas tank size will bring you across the desert
  - You can convince yourself that this function is monotonic
  - The bounds are well defined (0 to 10000)
  - $\log_2(10{,}000{,}000)$

Through the Desert

```
public static final double EPS = 1e-9;

boolean can(double f) {
  // return true if your jeep can reach goal state with fuel tank
  // capacity f, return false otherwise
}

// inside int main(), binary search the answer, then simulate

double lo = 0.0, hi = 10000.0, mid = 0.0, ans = 0.0;
while (Math.abs(hi - lo) > EPS) { // while the answer is not found
  mid = (lo + hi) / 2.0; // try the middle value
  if (can(mid)) {
    ans = mid; hi = mid; // save the value
  } else {
    lo = mid;
  }
}

// after the loop above is over, we have the answer
System.out.printf("%.3lf\n", ans);
```

Bitmask problem

- Find the minimum value x between L and U such that x OR M is maximized
- $0 <= U, L, M < 2^{31}, L <= U$

Bitmask problem

- Find the minimum value x between L and U such that x OR M is maximized
- $0 <= U, L, M < 2^{31}, L <= U$
- Observations:
  - The first goal is to maximize x OR M

Bitmask problem

- Find the minimum value x between L and U such that x OR M is maximized
- $0 <= U, L, M < 2^{31}, L <= U$
- Observations:
  - The first goal is to maximize x OR M
  - The second goal is minimize x

Bitmask problem

- Find the minimum value x between L and U such that x OR M is maximized
- $0 <= U, L, M < 2^{31}, L <= U$
- Observations:
  - The first goal is to maximize x OR M
  - The second goal is minimize x
  - x should resemble ~x because x OR ~x = 111..1

Bitmask problem

- For the *i*th bit from 31 → 0, set $x' = x \mid 2^i$ if

  1) $x' <= U$ and $M \& 2^i = 0$

  OR

  2) $x' <= L$

Bitmask problem

- For the *i*th bit from 31 → 0, set $x' = x \mid 2^i$ if

  1) $x' <= U$ and $M \& 2^i = 0$

  OR

  2) $x' <= L$

- Condition 1 guarantees *x* remains under *U* (maximally) and is not needlessly set (minimizes)

Bitmask problem

- For the *i*th bit from 31 → 0, set $x' = x \mid 2^i$ if

  1) $x' <= U$ and $M \& 2^i = 0$

  OR

  2) $x' <= L$

- Condition 1 guarantees *x* remains under *U* (maximally) and is not needlessly set (minimizes)
- Condition 2 guarantees *x* is at least *L*

Bitmask problem

- Challenge problem!! Not at all obvious
  - An odd mix between binary and decimal properties
- Technically a binary search :)

More on binary search

- TopCoder article

Practice

- Using the UVa system
  - I have the following people's user IDs
    - Brett – brett1479
    - Ben – Benjacat
    - Shohan – Xubes
    - Bowen – yubowenok
    - Kyle – coronelka
    - Aviv – arg450
    - Avi – stimpy
    - Weilan – weilun_du
    - Teddy – tkatz
  - if your name is not on the list see me

For next class

- Readings:
  - Sections 3.3 and 3.4
- Exercises:
  - Catch up on your homework
    - Due next class
  - Pick a midterm problem
    - Due next class