Alberta Collegiate Programming Contest

2017

Not the easiest ACPC, but hope you had fun!

Statistics

Problem	Modan’s	Wenli’s	Actual
Anthony and Diablo	100%	100%	90%
Musical Scales	90%	60%	75%
Tetration	50%	5 teams	65%
Divide by 100...	70%	60%	60%
Quantum Superposition	10%	3 teams	10%
Concentration	40%	75%	5%
Lane Switching	15%	4 teams	1
Race Track	10%	1 team	0
Maximal Sequence	< 2 teams	0 teams	0
Matchings	< 2 teams	0 teams	0

Problem	Difficulty
Anthony and Diablo	2.1
Musical Scales	2.0
Tetration	2.6
Divide by 100...	5.7
Quantum Superposition	8.7
Concentration	7.7
Lane Switching	8.9
Race Track	7.5
Maximal Sequence	9.3
Matchings	9.0

Solutions

Not going to go over details or take questions.
Come to the next CPC meeting!

Anthony and Diablo

- Trivial implementation
- Author: Modan

Anthony and Diablo

Observe optimal shape is a circle.

Check max area created by fence is at least A,

```
maxA = (N*N)/(4.0*math.pi)
```

```
print("Diablo is happy!" if maxA >= A else "Need more materials!")
```

or required amount of fencing is at most N.

```
    // pi r r = a
    double r=sqrt(a/pi);
    // pi 2 r = c
    double c=pi*r*2;
    cout<<(c>n?"Need more materials!":"Diablo is happy!")<<endl;
```

Musical Scales

- Implementation, brute force
- Author: Zachary Friggstad

Musical Scales

There are only 12 musical scales and input size is small; just brute force.

For every major, check if it is possible for the song.

To check if a note is in a major, use simple “distance” function

```
vector<int> p={0,2,4,5,7,9,11};

bool in(int note, int major){
    for(int i:p)
        if((major+i)%12==note)return 1;
    return 0;
}
```

Tetration

- Math
- Author: Modan

Tetration

Come up with the following theorem (or just find pattern, sample cases are big hints!).

$$^{\infty}a = N \text{ iff } a^N = N\text{, therefore } a = N^{1/N}.$$

Solution is easy if you noticed this, and much harder if you used other approaches.

```
print("%.6f"%(n**(1/n)))
```

Or use binary search, however, it’s not obvious how many iterations is good enough for approximating infinite tetration. In Python, 20 iterations of binary search and 250,000 iterations of exponentiation is good enough for 6 digits of precision, and runs under a second.

Divide by 100...

- Implementation, Strings
- Author: Modan

Divide by 100…

Python division/Java BigInteger doesn’t work… Not enough precision/arithmetic is too slow.

Observe division by a “special” number can be treated as a string operation, which is much more efficient than doing arithmetic on numbers up to 10^(10^6).

Insert decimal point in the middle or in front of N, depending on the *length* of N and M.

Watch out for edge cases, e.g. inserting leading zeroes, removing trailing zeroes, removing the decimal point.

Quantum Superposition

- Dynamic programming
- Author: Wenli Looi

Quantum Superposition

Let f:V->{Z} be a function, f(v) is the set of distances to reach vertex v.

Observe the recurrence

$$f(v) = \bigcup f(u)+w, \text{ where for all } u,w, (u,v,w) \text{ is an edge.}$$

And base case is f(root)={0}.

Naive brute force is too slow, use memorization/DP!

For each query q, check if there exists element e in graph 1’s set, such that the element (q-e) exists in graph 2’s set. Print “Yes” if so, and “No” otherwise.

Concentration

- Simulation, implementation
- Author: Modan

Concentration

Observe at any given time at most 1 pair of “similar” cards is known (due to the amazing plays by Anthony and Matthew); memorize this pair.

Memorize all revealed card; this can be done with an array. Memorize each player’s position in their “random” choice array. Memorize current player.

Simulate the game. If (known “similar” pair is not null), add points to the current player, grant extra turn; reveal next “random” card chosen by the current player and switch player otherwise.

Compare points when the game ends.

Lane Switching

- Graph
- Author: Howard Cheng, Brandon Fuller

Lane Switching

Construct weighted graph; the vertices are unoccupied spaces; an edge (x,y,w) exists iff x and y are in adjacent lanes, and there are enough space to switch lanes from x to y, the weight w is the safety factor of y.

Use modified Prim's-like (or Dijkstra's-like) traversal; the "weight" or "distance" of a traversal is the current max edge weight/safety factor.

Race Track

- Geometry
- Author: Modan

Race Track

Not difficult, lengthy implementation.

For each corner, slice the edges by the smoothing length, and create an arc object. Exact implementation can vary quite heavily.

For each query, compute the min distance to all the sliced line segments; compute point to line segment distance. Compute the min distance to all arcs; if the point is in the angle of the arc, return the distance to the circle of the arc; return the min distance to one of the endpoints of the arc otherwise.

Return the overall min distance.

Maximal Sequence

- Segment tree
- Author: Zachary Friggstad

Maximal Sequence

Create a segment tree for the original main sequence that stores for various intervals [i, j) a set of all notes in that range. The total space is O(n log n) as there are log n levels of the tree. Using an unordered set, it takes about O(n log n) time to construct as well.

Now for each query (i, B), have a recursive function that given an interval [lo,hi) into the original sequence will compute the longest sequence that can be played within the interval [lo,hi) starting at max(i,lo). If the set of notes in the sequence [lo, hi) is a subset of B, then just return hi-max(i,lo). This takes O(|B|) time using. Otherwise, you have to do some recursive calls.

Each O(|B|) subset query will be done O(log n) times.

Matchings

- Fast Fourier Transform
- Author: Kent Williams-King

Matchings

Naive brute force is too slow. Let N be the size of the floor, brute force runs in O(N^2), i.e. 10^12 operations, which is too slow.

Use Fast Fourier Transform!

FFT multiplies two polynomials of size N in O(N log(N)) time.

Matchings

a_1	a_2	a_3	b_1	b_2	
a_4	a_5	a_6	b_3	b_4	
a_7	a_8	a_9			

Consider:

$$(a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4 + a_6x^5 + a_7x^6 + a_8x^7 + a_9x^8) \times (b_4 + b_3x + b_2x^3 + b_1x^4)$$

The coefficient of the x^4 term is: $a_1b_1 + a_2b_2 + a_4b_3 + a_5b_4$
The coefficient of the x^8 term is: $a_5b_1 + a_6b_2 + a_8b_3 + a_9b_4$
The coefficient of the x^3 term is: $a_1b_2 + a_3b_3 + a_4b_4$ (means nothing)
etc.

Big thanks to everyone who helped working on this contest! Including:

Zachary Friggstad, Coach of UofA

Howard Cheng, Coach of UofL

Brandon Fuller, UofL alumni

Wenli Looi, Kent Williams-King, UofC alumni