NCPC 2008
Presentation of solutions

Nils Grimsmo

2008-10-04

E - Event Planning

Problem

Given a budget B, N participants, H hotels to choose between, and W different weekends you can go, find the cheapest hotel with enough availability.

Solution

```
minimum := B + 1
for h := 1..H:
    if N * price[h] < minimum:
        for w := 1..W:
            if availability[h][w] >= N:
                minimum := N * price[h]
if minimum <= B:
    print minimum
else:
    print "stay home"
```

B - Best Compression Ever

Problem

Given N different files, is it theoretically possible to store them as compressed files of length at most b? Can theoretically have contents of all files in de-compressor program source code, but must be able to distinguish all the files. So the question is how many unique files of length at most b bits can you have?

Observations

- Can have compressed files of lengths $0, \ldots, b$.
- 2^b unique files of length b, 2^{b-1} of length $b-1$, etc..
- $\sum_{i=0..b} 2^i = 2^{b+1} - 1$

Implementation

```
if ((1LL << (b+1)) - 1 >= N ) print "yes"
else                          print "no"
```

G - Getting Gold

Problem

Simulate the players walk through the maze, never moving to an unexplored square if sensing a draft. Count the number of gold pieces you can safely pick up.

Solution

Procedure find(y,x):

- If square is wall or marked visited: return 0
- If square has gold: gold = 1, else: gold = 0
- Mark square visited
- If some neighboring squares has trap: return gold
- gold += find(y-1,x) + find(y,x-1) + find(y+1,x) + find(y,x+1)
- return gold

Call find(y,x) with y,x for starting position

A - Aspen Avenue

Observation

If you move the trees in order from the first to the last, it will always be best fill the first free hole on either the left or right side.

Recursion

Cost with l trees left to place on the left and r on the right:

$$C(l,r) = \begin{cases} 0 & \text{if } l + r = 0 \\ \min \begin{cases} C(l-1,r) + \text{dist. left hole} & \text{if } l > 0 \\ C(l,r-1) + \text{dist. right hole} & \text{if } r > 0 \end{cases} & \text{otherwise} \end{cases}$$

Implementation

- Sort the trees on drop position.
- Memoize or use DP to avoid exponential run-time from recursion.

C - Code Theft

Problem

- If the lines in the fragments are viewed as symbols, the task is to find the length of the *longest common substring* (in this string of lines) between each fragment and the new snippet, reporting the fragments for which it is maximal.
- Strings both in the repository of code fragments, and in the new snippet, must have leading and trailing space removed, and other multiple spaces normalized to one space.

Naïve solution

- For each code fragment:
- For each line in the fragment:
- For each line in the new snippet:
- Match as many lines downward as possible

$\mathcal{O}(N \cdot n^3 \cdot \mathrm{m})$, n maximal fragment length, m maximal line length.

C - Code Theft - continued

Speed ups

- Avoid string compare:
  - Hash all strings once to speed up comparison.
  - or sort all strings, extract unique, and use order (loose $\mathcal{O}(m)$ factor).
- If you already have compared starting at position j in a fragment and position k in the new snippet, and found a match of say 7 lines, there is no point in a comparison starting at $j + 1$ and $k + 1$, which will give 6 lines.
  - Try all $\mathcal{O}(n)$ *alignments* of repository fragment and new snippet, checking consecutive aligned lines for match.

New running time: $\mathcal{O}(N \cdot n^2)$.

C - Code Theft - continued

Optimal solution

- Build a *suffix tree* for the string of lines (string identifiers) from the new snippet.
- For each fragment from the repository, find its *matching statistics*.

Running time: Linear in the total number of input characters.

I - Introspective Caching

Problem

Given a cache of size c, n different objects in total, and a object accesses known in advance, how many times do you have to read an object into cache, given an optimal strategy?

Solution

Maintain the following:

- The list of accesses (mapping from time to object).
- For each object, the list of remaining access times in order.
- The set of cache resident objects.
- A sorted mapping from first upcoming access time, to which object, but only for resident objects.

When you need to evict something, use the sorted mapping to find the resident object which will be needed again last. Update all data structures when taking things in and out of the cache.

D - Dinner

Problem

Given a group of m people, is it possible to pick a (smallest) year Y, and divide the group into two parts, such that every pair in the first group (1) met before Y, while every pair in the second group (2) met in or after year Y? None of the two parts must be larger than $\frac{2}{3}n$.

D - Dinner - continued

Solution

Try all years Y. For every person, count the number of others he met before (m_b), and in/after (m_a) year Y ($m_b + m_a = m$). You can sometimes decide directly whether a person goes in 1 or 2:

- $m_b + 1 < \frac{1}{3}m \Rightarrow$ he cannot be in 1.
- $m_a + 1 < \frac{1}{3}m \Rightarrow$ he cannot be in 2.

From there, we'll just have to try and fail:

- Put person in 1 $\Rightarrow$ put those met in/after Y in 2.
- Put person in 2 $\Rightarrow$ put those met before Y in 1.

Branch and bound, trying the alternative eliminating the most first.

H - Hard Evidence

Problem

Given a convex polygon of n vertices, and a circle of radius r strictly enclosing it, find the maximal view angle of the polygon from a point on the circle.

Solution

- For all possible pairs of vertices i, j:
    - Search for the position p on the circle where the angle between the lines $p - i$ and $p - j$ is maximal.
    - Start at the first and last position from which the line segment is visible.
    - Do a golden bound search (trinary search) for a maximum, as the view angle function in now strictly convex.

This of course involves some trigonometry...

F - Fixing the Bugs

Problem

Given is T hours, and B bugs to fix, where you work on one bug each hour. Each bug has a severity s_i and a fix probability p_i. If you fail on fixing a bug, it's fix probability now becomes $p_i \cdot f$, where $0 \leq f \leq 1$ is a fixed measure of your self esteem.

Recursion

$$S(T, p_1, \cdots, p_B, s_1, \cdots, s_B)$$
$$= \max_{1 \leq i \leq B} \begin{cases} 0 & \text{if } T = 0 \\ p_i * (s_i + S(T-1, \dots, 0 \cdot s_i, \dots) & \text{otherwise} \\ +(1-p_i) * S(T-1, \dots, f \cdot p_i, \dots) & \end{cases}$$

F - Fixing the Bugs - continued

Observations

Of course, running DP on all these variable doesn't work, but:

- You can greedily make a local choice to fix the bug with maximal current $p_i * s_i$ (proof omitted...).
- You can do some clever DP:
  - Given a time left, a set of unfixed bugs, and a number of failures on unfixed bugs, you will always do the same choices.
  - Therefore you can run DP on these three, where the second is a bit pattern.
  - Keep the p_i's updated as you recurse with failures

If you also do DP on the greedy local choice, you get a running time of $\mathcal{O}(T^2 2^B)$.

J - Just a Few More Triangles

Problem

Given an integer n, find the number of triples a, b, c such that $a^2 + b^2 \equiv c^2 \mod n$, where $1 \leq a \leq b < n$ and $1 \leq c < n$.

Solution

- n to large for naïve solution.
- Need Chinese remainder theorem or FFT, pluss some tricks...
- Update slides may come later...