COPYRIGHT POLICY

版权声明

BITTIGER #16.1

Dec 3 / 2016

PROBLEMS

Gas Stations ☆☆☆

- Dynamic Programming, Greedy, Binary Search

Logic Expression Tree ☆☆☆☆

- Dynamic Programming, Binary Tree

Keywords Filter ☆☆☆☆

- String, Trie Graph

Text

GAS STATIONS

谷歌**google**跪经

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=212722&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26sortid%3D311

印度小哥，全程没有提示。这道题完全不知道怎么写，最后瞎写了一下。。。

题目：有一条公路，长度是m, 中间有k个加油站，由此我们可以得到一个加油站之间的最大距离，然后给你一个数t，这个数代表增加的加油站的数量（往里面插入），求使得加油站之间最大距离变得最小的值，返回这个最小的最大距离

GAS STATIONS

http://hihocoder.com/contest/hihointerview25/problem/1

N Gas stations

- All on Axis-X
- **$0 = X_1 \leq X_2 \leq \ldots \leq X_N = M$**

You can build K more stations

- For a construction plan P
- $S(P)$ = maximum distance between two adjacent gas station

Calculate $\min\{S(P)\}$

GAS STATIONS

Input:

- N, M, K
- $X_{1\sim N}$

Output:

- $\min\{S(\boldsymbol{P})\}$

BASIC THINKING

Only 2 stations

- Averagely deploying new stations

Enumerate number of stations between 2 adjacent old stations

- $\frac{X_i - X_{i-1}}{1+K_i}$ for each segment
- $\min\{\max\{\frac{X_i - X_{i-1}}{1+K_i}\}\}$
- $\boldsymbol{O(K^N)}$

Total enumeration is in multi-steps

STEPS

After enumerated $X_1 \sim X_i$

- **Number of new stations between each adjacent old stations is not important !**
- A **state** can be represented by
  - **Number of old stations I already enumerated** – Process of enumeration
  - **Number of new stations used** – How many I can use now
  - **The maximum distance now exists** – How I calculate the $S(P)$

$\boldsymbol{F(i,j)}$

- After enumerated $X_1 \sim X_i$, used j new stations.
- **The minimum of maximum distance We can get.**

$\boldsymbol{F(i,j)}$

- $= min\left\{max\left\{F(i-1,j'), \frac{X_i - X_{i-1}}{1+j-j'}\right\}\right\}$

DYNAMIC PROGRAMMING

Algorithm

- **Step 1:** Set state $F(i, j)$
  - After put j new stations between $[X_1, X_i]$, the minimum $\max\{X'_i - X'_{i-1}\}$
- **Step 2:** Calculate state from $i = 2 \sim N, j = 0 \sim K$
  - $\boldsymbol{F(i,j) = min\left\{max\left\{F(i-1, j'), \frac{X_i - X_{i-1}}{1+j-j'}\right\}\right\}}$
- **Step 3:** Output $F(N, K)$ as answer

Time Complexity:

- $O(NK^2)$
- Can't solve all of the data

CODE EXAMPLE

```
f[1][0]=0;
for i=2~N
    for j=0~K
        for j2=0~j
            f[i][j]=min(f[i][j], max(f[i-1][j2], (x[i]-x[i-1])/(1+j-j2)));
Output(f[N][K]);
```


Text

BETTER DP?

Like $O(NK)$

It's hard to modify the $O(NK^2)$ Algorithms

- Common way is to make state $F(i, j)$ only calculated by $F(i-1, j-1)$
- But it's hard to do such thing here
  - Since the calculation of "cost" require $(j - j2)$

We need to change our way of thinking !

STILL STEPS

If we only have 1 new station, where will you put it ?

- The maximum distance between 2 adjacent old stations.
- Or the answer won't change.

If we only have 2 new stations, where will you put them ?

- The position of 1st new station won't change.
  - $X_i \sim X_{i+1}$
  - At least 1 new stations will be put there, or the answer won't change.
  - Why don't make it the 1st one.
- The position of 2nd new station is similar to the 1st .
  - Only differs when we try to put it at $X_i \sim X_{i+1}$
  - It's not like we put new stations twice.
  - It's we put them together.

What if we only have K new stations ?

GREEDY

Algorithm

- **Step 1:** Use $Count(i)$ to maintains number of new stations between $[X_i, X_{i+1}]$
- **Step 2:** Enumerate $j = 1 \sim K$ to deal with each new station separately.
  - **Step 2.1:** Find the maximum $\frac{X_{i+1}-X_i}{Count(i)+1}$ as i', by maintaining a max-heap H.
  - **Step 2.2:** $Count(i') + 1$, maintain the max-heap H.
- **Step 3:** Output H's top as answer

Time Complexity:

- $O(\log N\, K)$
- Solve all of the data

NO MORE STEPS

We try almost all ways from input=>output direction.

- Another way is to reverse it.

If we require Answer to be less than Ans

- What's the minimum K?
- $Rf(Ans) = K$
  - $= \sum ceil\left(\frac{X_i - X_{i-1}}{\text{Ans}}\right) - 1$

$$Rf(Ans_1) \geq Rf(Ans_2), for\ Ans_1 < Ans_2$$

If we find the minimum Ans, that $Rf(Ans) \leq K$

- Ans will be our Answer

BINARY SEARCH

Algorithm

- **Step 1:** Limit range of searching $[L, R]$ to $(0, M]$
- **Step 2:** Keep choosing an answer ans from center of the range $[L, R]$
  - If $Rf(Ans) \leq K$, set R to Ans
  - If $Rf(Ans) > K$, set L to Ans
  - Until $(r - l)$ is very small
- Step 3: Output l as Answer

Time Complexity:

- $O(N \log K)$
- Solve all of the data

CODE EXAMPLE

```
double l = 0, r = 100000;
while (l + 1e-4 < r) {
        double mid = (l + r) /  2;
        int sum = 0;
        for (int i = 1; i < n; i++) sum += ceil(1.0 * (a[i] - a[i - 1]) / mid) - 1;
        if (sum > k) l = mid;
        else r = mid;
}
printf("%.1lf\n", l);
```

Text

THINKING PATH

Brute force

- Too much complexity

Multi-Step

- Consider by old-stations => Dynamic Programming $O(NK^2)$
- Consider by new-stations => Greedy $O(\log N\ K)$

Reverse more - Consider by answers

- Monotonicity => Binary Search $O(N \log K)$

QUESTION TIME

THANKS FOR LISTENING