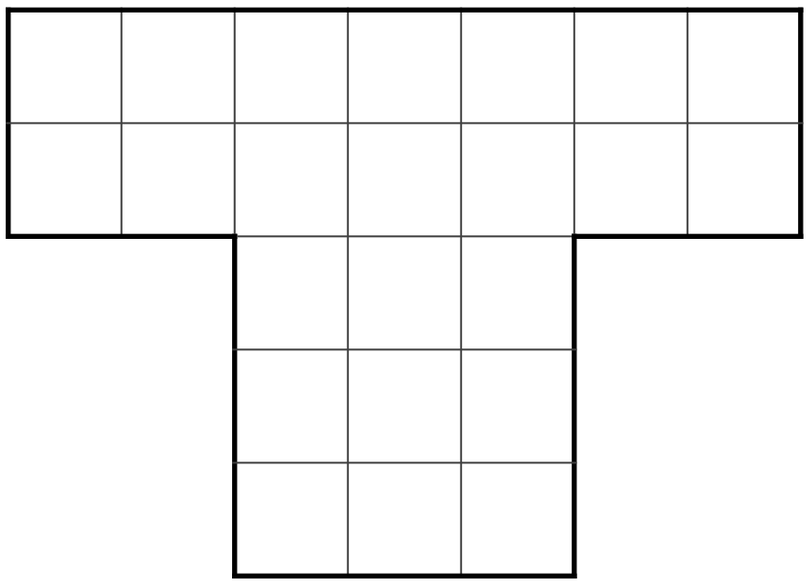
C：壺

@Yazaten

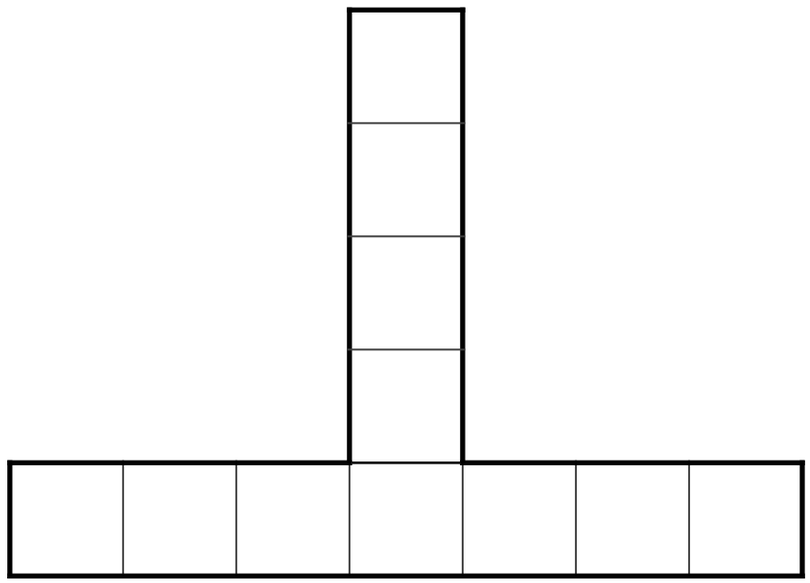
問題概要

変わった形の N 個の壺に体積 M の水を注ぎ
それぞれの水面の高さの和の最大値を求める。

N=2 , M=15



壺0

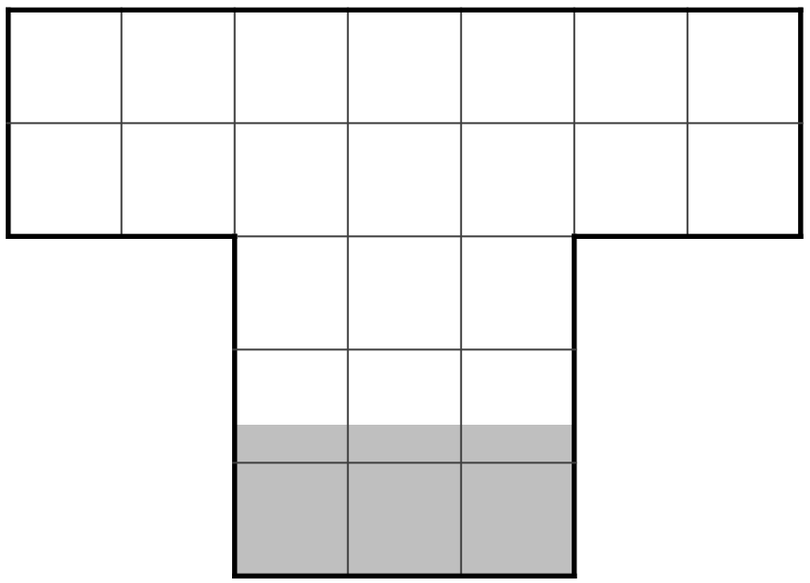


壺1

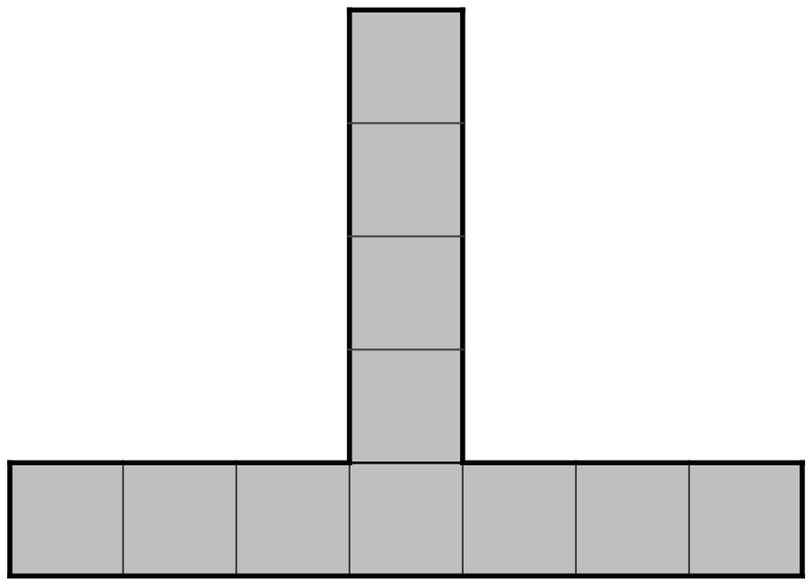
問題概要

変わった形の N 個の壺に体積 M の水を注ぎ
それぞれの水面の高さの和の最大値を求める。

N=2 , M=15



壺0



壺1

考察

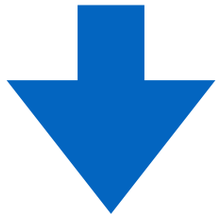
底面積が等しい壺が複数ある問題 であれば貪欲法で解ける

全ての壺の中で水面を高くするための効率が良い物に
水を入れていく貪欲で解ける？

考察

底面積が等しい壺が複数ある問題 であれば貪欲法で解ける

全ての壺の中で水面を高くするための効率が良い物に
水を入れていく貪欲で解ける？



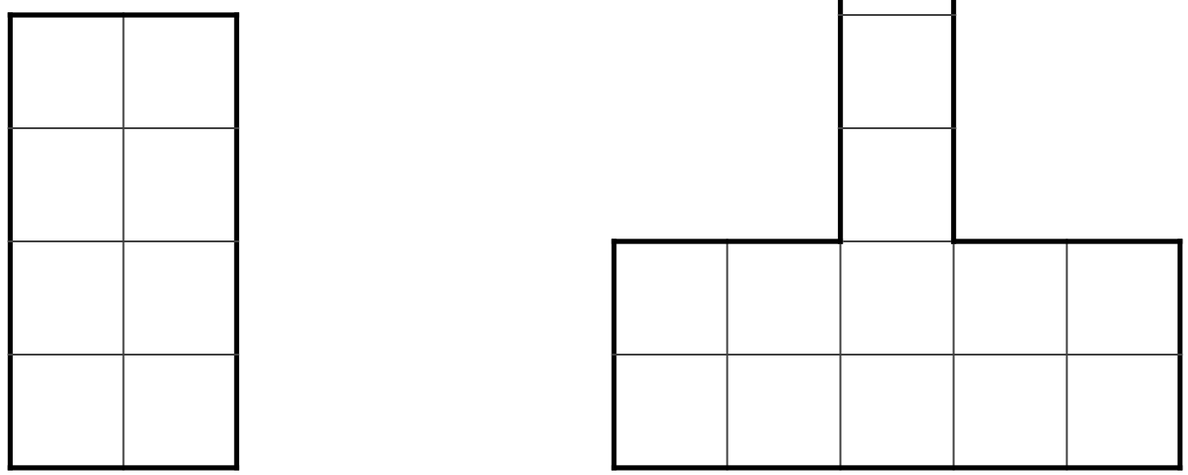
解けない!!

考察

効率の悪い円柱に水を注ぎ、その上の効率の良い円柱に水を注ぐのが最善となるケースが存在する

例

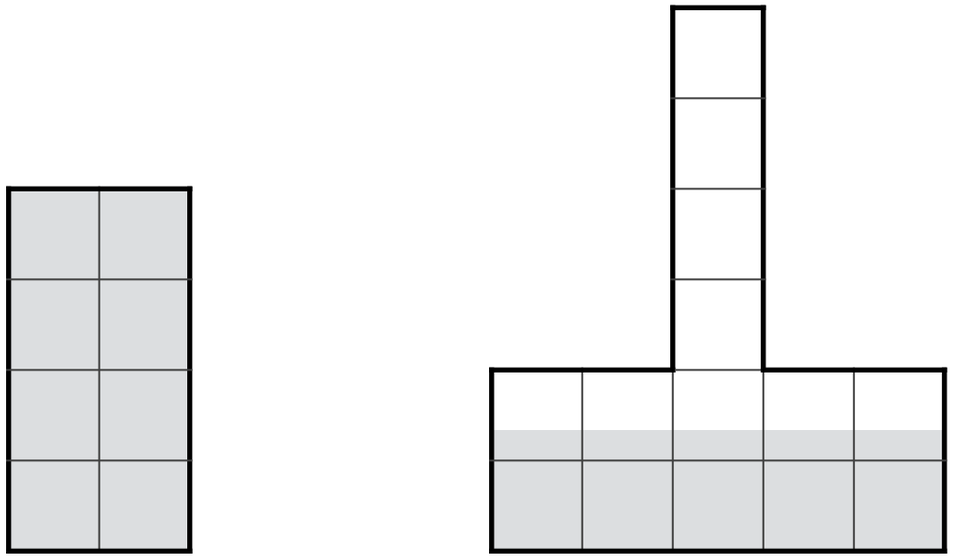
N=2 , M=14



考察

×

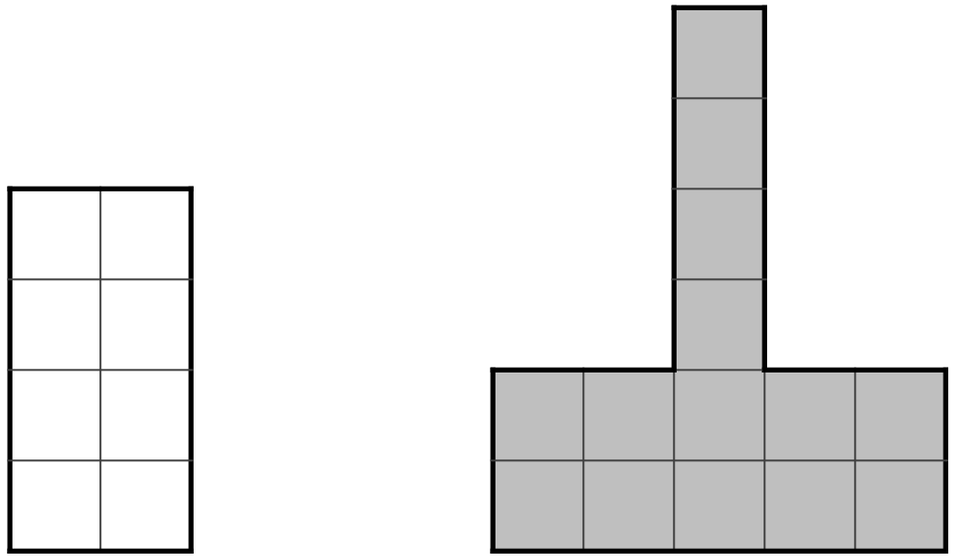
貪欲



高さの総和 -> 5.2

○

最適解



高さの総和 -> 6

考察

貪欲法では解けないので それぞれの壺に入れる
水の量の組み合わせを全探索する必要がある。

愚直に全探索 -> O((SHK)^N)

最悪のケースだと計算量が 8000の200乗　とかでヤバい

考察

動的計画法を使って効率的に計算してやればOK

DPで効率的に全探索 -> O(NM^2)

N=M=200 なので間に合う

解法

全ての壺に 水を合計 M 注いだ場合のみを
全探索すれば良い。

そのため、後述のDPで解くことが出来る。

解法

① 状態の持ち方

dp[i][j]

= i番目の壺までに水をj入れた時の高さの総和の最大値

(dpの初期値0)

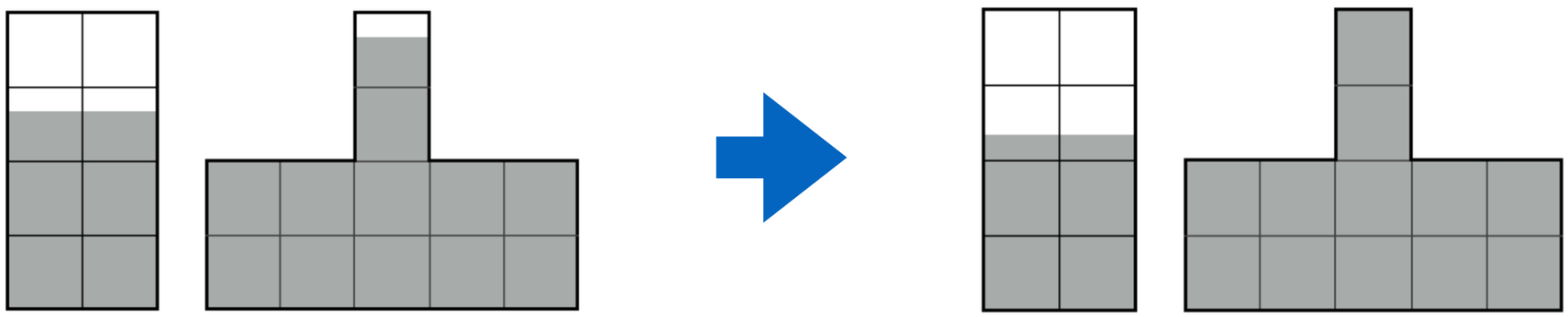
解法

① 状態の持ち方

水の高さが整数でない壺 (以下 中途半端な壺) が1つ以下となるような
最適な注ぎ方が必ず一つ以上存在するため、jは整数値で考えれば良い。

理由を以下の中途半端な壺が2つ存在する例を用いて説明する。

このようなケースでは、一方の壺の水を他方に注ぎ変えることで、
必ず同等またはより良い解にすることが可能である。
中途半端な壺が一つしか存在しないように整数量の水を注ぐと
壺に注がれる水は必ず整数量になり、jが整数値で良いことがわかる。



解法

② 遷移の仕方

dp[i][j] = max(dp[i-1][j-k]+l[i][k] , dp[i][j])

の漸化式が成り立つ

※l[i][k]はl番目(1-index)の壺に水をk入れた時の水面の高さ

解法

② 遷移の仕方

dp[i][j] = max(dp[i-1][j-k]+l[i][k] , dp[i][j])

の漸化式が成り立つ

i-1番目までの壺に合計j-kの水を入れた時の水面の高さの総和

+

i番目の壺に合計kの水を入れた時の水面の高さ

解法

② 遷移の仕方

dp[i][j] = max(dp[i-1][j-k]+l[i][k] , dp[i][j])

の漸化式が成り立つ

i-1番目までの壺に合計j-kの水を入れた時の水面の高さの総和

+

i番目の壺に合計kの水を入れた時の水面の高さ

解法

実装例

```
double l[200][8000]={};
double dp[201][201]={};
int main(){
    int n,m,k,s,h;
    rep(i,MAX_N)rep(j,MAX_K*MAX_H)l[i][j] = -1;
    rep(i,MAX_N)l[i][0]=0;
    cin>>n>>m;
    rep(i,n){
        cin>>k;
        int filled=0;
        rep(j,k){
            cin>>s>>h;
            for(int k=1;k<=s*h;k++) l[i][filled+k] = l[i][filled+k-1] + (1.0/s);
            filled += s*h;
        }
    }
    rep(i,MAX_N+1)rep(j,MAX_M+1)dp[i][j]=0;
    for(int i=1;i<=n;i++)
        for(int j=0;j<=m;j++)
            for(int k=0;j-k>=0;k++)
                if( dp[i][j] < dp[i-1][j-k]+l[i-1][k] + 1e-9 )
                    dp[i][j] = dp[i-1][j-k]+l[i-1][k];
    printf("%.10lf\n",dp[n][m]);
}
```

解法

同じDPでも、水面の高さを1づつ増やしていくDPでは
間違ってしまうので注意
(総和の最大値が整数でない場合にダメ)

👇間違ったDPの状態と遷移

dp[i][j]

= i番目の壺までの水面の高さの総和をjとするために必要な水の量

dp[i][j] = max(dp[i-1][j-c[i-1][k]]+k , dp[i][j])

実装上の注意など

事前に l[i][k] = l番目(1-index)の壺に水をk入れた時の高さ
という配列を事前に用意しておくと実装が非常に楽

誤差が発生する場合があるので 小数点以下5桁以上
出力する必要がある

ジャッジ解

Yazaten　　:　42行

Respect2D　:　76行

T.M　　:　29行

tubo28　　:　68行

総評

AC/Submit : 65.38 %

FA-onsite : KUROGANE 51min

FA-online : asi1024 22min