第4回 JOIG 春トレ Contest2

チョコレートを増やせ解説

担当：Thistle（チューター）

目次

問題概要

チケット × Ai ⇒ (チョコレート+チケット) × Bi

という交換規則が N 個ある。

最終的に K 個以上のチョコレートを手に入れるためには、

最初にチケットとチョコレートを最低何個ずつ購入する必要があるか？

1≦K≦M について求めよ。

問題概要

サンプル 1

規則 1 : TICKET x 3 ⇒ (+ TICKET) x 1
規則 2 : TICKET x 5 ⇒ (+ TICKET) x 2

基礎方針

考察

答えの代わりに、1 以上 M 以下の全ての x について

x 個のチョコレートを買った時の
最終的に得られるチョコレートの最大数 y

を求めることにする。

重要な考察　y 個未満を得る行動は考えないでよい。

⇒ 全ての x について y = f(x) が一意に定まる。

基礎方針

考察

全ての x について y = f(x) が一意に定まる。

⇒ **ちょうど** y 個のチョコレートを
得るときに必要な最小の x が求まる。

⇒ y 個**以上**のチョコレートを得るときに必要な
最小の x が求まる。これは求めたい値そのもの。

基礎方針

考察

買	得
1	1
2	3
3	3
4	5
5	6

丁度

得	答
1	1
2	-
3	2
4	-
5	4

以上

得	答
1	1
2	2
3	2
4	4
5	4

小課題 1

制約	M≦500, N=1

解法

全ての購入数 x について、使えるだけ規則を使っていけば答えが得られる。

計算量は O(M*M)

小課題 2

制約	M≦500, Ai- Bi=const.

解法

一度規則を使った後チケットが減る枚数 Ai-Bi が同じ
⇒ できるだけ **多いチョコを得られる規則を使うほうが得**
⇒ 使えるチケット（現在のチケット枚数 ≧ Ai であるもの）
　の中で最大の Bi を持つ規則を使う**貪欲法**を行う。
計算量は O(M*M*N)

小課題 3

制約	M≦500, N≦500

考察

小課題 1,2 みたく貪欲法をしたいが、どうにも動かなそう。
こういった場合は、DPや賢い探索など他の解法を探さないといけない。
今回は「使用枚数が x **枚を超えないように** Ai-Bi 枚のチケットを**使っていき、Bi を最大にする**」という考察から、
ナップサック問題に見立てて解く。

小課題 3

解法

全ての x について、変則的なナップサック問題

容量 x のかばんに入る荷物の価値の合計の最大値は何か？

荷物 i の重さ = Ai-Bi，荷物 i の価値 = Bi

ただし　同じ荷物は何度でも使える。

現在の重さ+Ai≦xである荷物しか使えない。

を解く。(価値の最大値+x) が y になり、O(M*M*N)

満点解法

制約	M≦100,000, N≦500

考察

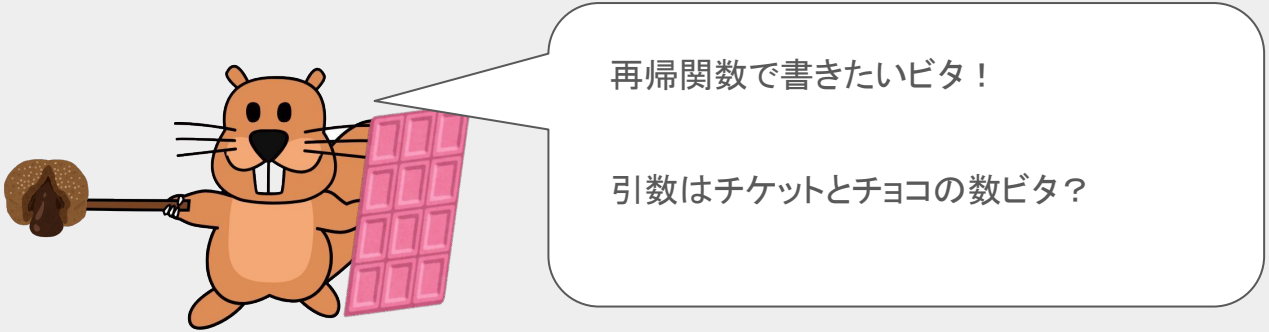
小課題 3 の解法は全ての M について行うと到底間に合わない。全ての M について同時にDPを行うことを考える。

とはいえどうやってDPすればよいのか？

満点解法

考察

DPには全探索の高速化という側面もあるので、全探索コードを書いてみると考察の参考になるかもしれません。
書いてみましょう。



満点解法

考察

再帰関数の引数は、チケットとチョコレートの数？

再帰関数の戻り値は、
今の引数で表される状態から始めて最後まで進めたときの結果
（今回はチョコレートの数の最大値）

⇒　実は、この値を求めるのに現在のチョコレートの数は必要ない！
⇒　引数はチケットだけで十分！

満点解法

考察

実際に再帰関数として実装してみると、以下のようになる。

```
int rec(int x) {
    int mx = 0;
    rep(i, n) if (x >= a[i].a) {
        chmax(mx, rec(x - a[i].a + a[i].b) + a[i].b);
    }
    return mx;
}
```

チケットが x 枚の時の最大値＝
(チケットが x-Ai+Bi 枚の時の最大値+Bi)の最大値

満点解法

解法

recは引数が同じなら同じ値を返すので、同じ引数のrecを2度呼び出さないようにしてみましょう（メモ化再帰）。

```
int dp[100005];
#define UNUSED -1
int rec(int x) {
    if (dp[x] != UNUSED) return dp[x];
    int mx = 0;
    rep(i, n) if (x >= a[i].a) {
        chmax(mx, rec(x - a[i].a + a[i].b) + a[i].b);
        if (mx > m) mx = m;
    }
    return dp[x] = mx;
}
```

状態数 M個

各rec O(N)

計O(NM) AC!!

別実装

解法

dp[x]はxが小さい順に更新されていくので、

dp[x]=x枚のチケットがある状態から得られる最大数として
dp[x]=max(dp[x-Ai+Bi]+Bi, dp[x])
としていく。
dp[x]+x≧y となる最小の x が ans[y] となる。

としても正しく更新できる。計算量 O(N*M) となり、AC!!

統計情報