響け！動的計画法(DP)

@motacapla

動的計画法(DP)とは

“ある問題を小さい部分問題に分割し, 部分問題の解を利用することで元の問題を解くアルゴリズム”

部分問題の解を「状態」とし, その状態の遷移を考える

様々なDP

問題参照先: AtCoder
解答URLを各スライドに記載しています

- 貰うDP
- 配るDP
- 確率DP
- bitDP
- 桁DP　今日はここまで！
- 期待値DP
- 区間DP
- 木DP (ツリーDP)
- ゲームDP
- 部分列DP
- 挿入DP
- 抜去DP
- 戻すDP
- 連結DP (フロンティア法)
- Trie DP
- 全方位木DP
- インラインDP
- Alien DP

(0-1)ナップサック問題

容量CのナップサックとN種類の品物(価値v_i, 重さw_i)が与えられた時,
Cを超えない範囲で価値総和を最大化するような品物の選び方を答えよ.

$1 \leq N \leq 100 \quad 1 \leq C \leq 10^5 \quad 1 \leq w_i \leq C \quad 1 \leq v_i \leq 10^9$ 2秒以内

例題: $C = 7$ $N = 4$

	価値	重さ
りんご	200円	3
ちくわ	150円	3
みかん	80円	2
もやし	80円	2

Q. (価値/重さ)順で貪欲に
足し合わせればいいのでは?
A. 最適解にならない

ナップサックに入った品物の価値テーブル

容量\品物	りんご	ちくわ	みかん	もやし
1	0円	0円	0円	0円
2	0円	0円	**80円**	80円
3	**200円**	200円	200円	200円
4	**200円**	200円	200円	200円
5	**200円**	200円	**280円**	280円
6	**200円**	**350円**	350円	350円
7	**200円**	**350円**	350円 →	**360円**

Educational DP Contest / DP まとめコンテスト　D - Knapsack 1 https://atcoder.jp/contests/dp/tasks/dp_d

二種類の記法

$dp[i][c]$:= iまでの品物の中から重さがcになるように選んだ時の価値の総和の最大値

- **漸化式**

https://atcoder.jp/contests/dp/submissions/4312994

$$dp[i][c] = \begin{cases} max(dp[i-1][c-w_i] + v_i, dp[i-1][c]) & \text{if 品物を選ぶ かつ } c - w_i \geq 0 \\ dp[i-1][c] & \text{if 品物を選ばない} \end{cases}$$

Pros. :

- 関数呼び出しによるスタックオーバーフロー†は発生しない

- インラインDP (セグ木で加算) のような高速化テクが適用できる

- **メモ化再帰**

https://atcoder.jp/contests/dp/submissions/7429375

```
Long long solve(long long i, long long c){
    if(dp[i][c] != -1) return dp[i][c];
    if(i == 0) return dp[i][c] = 0;
    if((c-w[i]) >= 0) return dp[i][c] =  max(solve(i-1,c-w[i])+v[i], solve(i-1,c));
    else return dp[i][c] = solve(i-1,c);
}
```

Pros. :

- トポロジカルソート‡不要

- コーナーケースでコードが簡素

*コーナーケース考慮のためのif文が増える: http://ideone.com/L3iZq

†再帰とスタック・オーバーフロー
http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/09-rec-iter.html

‡トポロジカルソート
https://en.wikipedia.org/wiki/Topological_sorting

貰うDP/配るDP

$dp[i][c]$:= iまでの品物の中から重さがcになるように選んだ時の価値の総和の最大値

- **貰うDP**

  https://atcoder.jp/contests/dp/submissions/4312994

  $dp[i][c]$へ状態を遷移

$$dp[i][c] = \begin{cases} max(dp[i-1][c-w_i] + v_i, dp[i-1][c]) & \text{if 品物を選ぶ かつ } c - w_i \geq 0 \\ dp[i-1][c] & \text{if 品物を選ばない} \end{cases}$$

- **配るDP**

  https://atcoder.jp/contests/dp/submissions/7510680

  $dp[i][c]$から状態を遷移

$$dp[i+1][c+w_i] = max(dp[i][c] + v_i, dp[i+1][c+w_i]) \quad \text{if } c + w_i \leq C \text{ (配り先が容量以内)}$$
$$dp[i+1][c] = max(dp[i+1][c], dp[i][c])$$

	…	i-1	i	i+1	…
c-wi		dp[i-1][c-w_i]			
c		dp[i-1][c]	dp[i][c]	dp[i+1][c]	
c+wi				dp[i+1][c+w_i]	

Educational DP Contest / DP まとめコンテスト　D - Knapsack 1 https://atcoder.jp/contests/dp/tasks/dp_d

確率DP

- **確率で状態を表すDP**

(例. dp[i] := ～となる確率)

N を正の奇数とします。

N 枚のコインがあります。 コインには $1,2,\ldots,N$ と番号が振られています。

各 $i\ (1 \leq i \leq N)$ について、コイン i を投げると、確率 p_i で表が出て、確率 $1-p_i$で裏が出ます。

太郎君は N 枚のコインをすべて投げました。

このとき、**表の個数が裏の個数を上回る確率**を求めてください。

入力はすべて整数である。$1 \leq N \leq 10^2$, $1 \leq a_1 < a_2 < \cdots < a_N \leq K$

- 方針と解答例

dp[i][j] := i枚目のコインまで投げたとき,
表がj枚でる確率

状態の遷移を考えると, 以下の式になる:

コインiが表の場合

$dp[i+1][j+1] \mathrel{+}= dp[i][j] *$ (コインiが表の確率)

コインiが裏の場合

$dp[i+1][j] \mathrel{+}= dp[i][j] *$ (コインiが裏の確率)

https://atcoder.jp/contests/dp/submissions/4298440

```
dp[0][0] = 1;
for(int i=0; i<n; i++) for(int j=0; j<i+1; j++){
  dp[i+1][j+1] += dp[i][j] * p[i];
  dp[i+1][j] += dp[i][j] * (1-p[i]);
}
double ans = 0.0;
for(int i=n/2+1; i<n+1; i++) ans += dp[n][i];
cout << setprecision(10) << ans << endl;
```

Educational DP Contest / DP まとめコンテスト I - Coins https://atcoder.jp/contests/dp/tasks/dp_i

bitDP

- **bitで状態を表すDP**

(例. dp[bit][i] :=)

- 制約が N <= 16 などと小さい時は大抵これ

N 匹のうさぎ(🐰1〜N)がいます。 これら N 匹のうさぎが徒競走をしました。

同着はいませんでした。 高橋君は M 人の観客から情報を集めました。

i 番目の観客によると、うさぎ x_i はうさぎ y_i よりも先にゴールしたそうです。

すべての**観客の情報に合致するような着順が何通り**考えられるか求めてください。

$2 \leq N \leq 16$, $1 \leq M \leq N(N-1)/2$, $1 \leq x_i, y_i \leq N$, $x_i \neq y_i$, (x_i, y_i)の組はすべて相異なる。

すべての観客の情報に合致するような着順が少なくともひとつ存在する。

例. $N = 4$ $M = 3$ うさぎ: 🐰

🐰1 🐰2 🐰3 🐰4

- 🐰1 は 🐰2 より先にゴール
- 🐰1 は 🐰3 より先にゴール
- 🐰4 は 🐰3 より先にゴール

パターン\着順	1位	2位	3位	4位
1	🐰1	🐰2	🐰4	🐰3
2	🐰1	🐰4	🐰2	🐰3
3	🐰1	🐰4	🐰3	🐰2
4	🐰4	🐰1	🐰3	🐰2
5	🐰4	🐰1	🐰2	🐰3

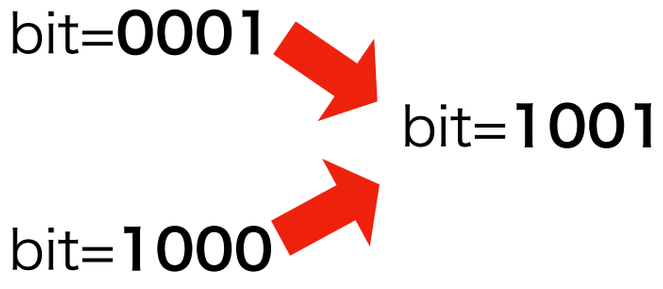
全部で5通り

AtCoder Beginner Contest 041 D - 徒競走 https://atcoder.jp/contests/abc041/tasks/abc041_d

bitDP: コード

- 方針と解答例
  - bitの各桁が🐰を表し, 1: 到着, 0: 未到着

    例. bit=1001 … 🐰1, 4が到着, 🐰2, 3は未到着
  - bit状態の遷移を考える



dp[bit] := 🐰到着順の通り数

https://atcoder.jp/contests/abc041/submissions/4184992

```
dp[0] = 1;
for(int bit=0; bit<(1<<n); bit++){ //bit=0000 から bit=1111まで
    for(int j=0; j<n; j++){
        if((bit>>j) & 1) continue;  //既に到着した🐰jはパス
        bool f = true;
        //g[j][k] := jはkより先に到着, 🐰kが既に到着していて🐰jが未到着ならfalse
        for(int k=0; k<n; k++) if(((bit>>k)&1) && g[j][k] == 1) f = false;
        if(f) dp[bit|(1<<j)] += dp[bit]; //例えば, bit=1001 に bit=0001 を足し合わせる
    }
}
cout << dp[(1<<n) - 1] << endl;
```

AtCoder Beginner Contest 041　D - 徒競走　https://atcoder.jp/contests/abc041/tasks/abc041_d

桁DP

- **条件を満たすか, 上限値か否か, 等の状態を表すDP**
  (例. dp[桁数][条件][未満or等しいのフラグ] := 場合の数)
- smallerフラグを設けて, 条件や上限の管理を行う
- 制約が K <= 10^10000 などと大きい時は大抵これ

1 以上 K 以下の整数のうち、十進表記における**各桁の数字の総和が D の倍数であるようなものは何個**でしょうか? $1e9+7$ で割った余りを求めてください。

$1 \leq K < 10^{10000}$, $1 \leq D \leq 100$

- 方針と解答例
  - K以下の整数らを先頭から1桁ずつ数字を見る
  - 各桁の数字の総和について、Dで剰余をとる
  - ある整数の数値がK以下ならば、加算する

例. K=123456, D=3

dgt=4の時

***?**

- …
- ***3** : dgt≦3(***部)が**123以下 (K以下)** なら加算する
- ***4** : dgt≦3が**123以下** かつ dgt≧5(**部)が56以下なら加算
- ***5** : dgt≦3が**122以下 (K未満)** なら加算
- …

Educational DP Contest / DP まとめコンテスト　S - Digit Sum https://atcoder.jp/contests/dp/tasks/dp_s

桁DP: コード

dp[dgt][d][isless] := 先頭から確定した桁dgt, 各桁の数字の和をDで割った値d†,
K未満/丁度のフラグislessを持つ整数の数

https://atcoder.jp/contests/dp/submissions/7505933

```
dp[0][0][0] = 1;
for(int dgt=0; dgt<N; dgt++) for(int d=0; d<D; d++) for(int isless=0; isless<2; isless++) {
 for(int nxt=0; nxt<10; nxt++) {
   if (nxt < K[dgt]-'0')          //ある整数の桁dgtがKのdgtより小さい場合 … isless=1に加算
     (dp[dgt+1][(d+nxt) % D][1] += dp[dgt][d][isless]%MOD)%=MOD;
   else if(nxt == K[dgt]-'0') //ある整数の桁dgtがKのdgtと同じ場合       … 各islessに加算
     (dp[dgt+1][(d+nxt) % D][isless] += dp[dgt][d][isless]%MOD)%=MOD;
   else                                //ある整数の桁dgtがKのdgtより大きい場合 … islessならば加算
     if(isless) (dp[dgt+1][(d+nxt) % D][isless] += dp[dgt][d][isless]%MOD)%=MOD;
 }
 //Kと等しい + K未満 - 1
 cout << ((dp[N][0][0] + dp[N][0][1])%MOD+MOD-1)%MOD << endl;
```

†(A+B)%C は A%C+B%C と等価
Educational DP Contest / DP まとめコンテスト　S - Digit Sum https://atcoder.jp/contests/dp/tasks/dp_s

ここまでのまとめ

- 動的計画法は「**ある問題を小さい部分問題に分割し, 部分問題の解を利用することで元の問題を解くアルゴリズム**」

- 記法は「**漸化式**」と「**メモ化再帰**」があるよ

- 種類は18種類以上、例として3つ挙げるよ:
  - bitDP : bitで状態を表すDP
  - 桁DP : 桁フラグで状態を表すDP
  - 確率DP : 確率で状態を表すDP

様々なDP

問題参照先: AtCoder
解答URLを各スライドに記載しています

- 貰うDP
- 配るDP
- 確率DP
- bitDP
- 桁DP
- 期待値DP
- 区間DP
- 木DP (ツリーDP)
- ゲームDP
- 部分列DP
- 挿入DP
- 抜去DP
- 戻すDP
- 連結DP (フロンティア法)
- Trie DP
- 全方位木DP
- インラインDP
- Alien DP

期待値DP

- **期待値で状態を表すDP**
(例. dp[i] := ～となる期待値)

N 枚の皿があります。 皿には $1,2,\ldots,N$と番号が振られています。

最初、各 i $(1 \leq i \leq N)$ について、皿 i には a_i $(1 \leq a_i \leq 3)$ 個の寿司が置かれています。

すべての寿司が無くなるまで、太郎君は次の操作を繰り返し行います。
$1,2,\ldots,N$ の目が等確率で出るサイコロを振り、出目を i とする。

皿 i に寿司がある場合、皿 i の寿司を 1 個食べる。 皿 i に寿司が無い場合、何も行わない。

すべての寿司が無くなるまでの操作回数の期待値を求めてください。

入力はすべて整数である。$1 \leq N \leq 300$, $1 \leq a_i \leq 3$

- 方針と解答例
  - 全皿は等確率で選ばれるので、a_iの順番は影響しない
  - 状態として皿数ごとの操作回数の期待値を考える

dp[i][j][k] := $a_i = 1$の皿数 i, $a_i = 2$の皿数 j, $a_i = 3$ の皿数 k における操作回数の期待値

https://atcoder.jp/contests/dp/submissions/4339992

```
double dfs(ll i, ll j, ll k){
    if(dp[i][j][k]>=0) return dp[i][j][k];
    if(i == 0 && j == 0 && k == 0) return 0.0;
    double ijk = (double)i+j+k; double res = n/ijk;
    if(k > 0) res += dfs(i, j+1, k-1) * (double)k/ijk;
    if(j > 0) res += dfs(i+1, j-1, k) * (double)j/ijk;
    if(i > 0) res += dfs(i-1, j, k) * (double)i/ijk;
    return dp[i][j][k] = res;
}
```

Educational DP Contest / DP まとめコンテスト　J - Sushi https://atcoder.jp/contests/dp/tasks/dp_j

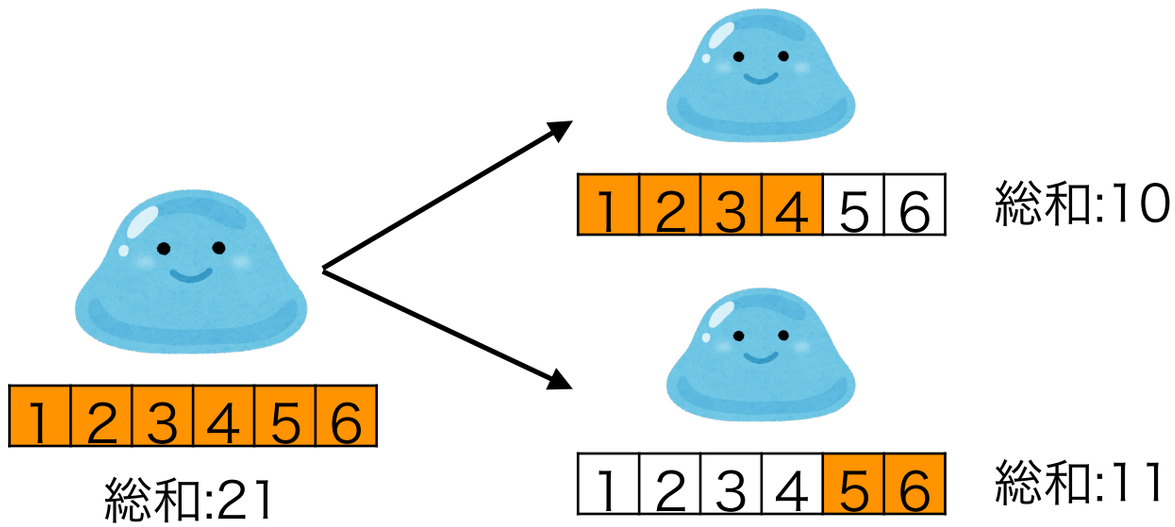
区間DP

- **左右の区間の状態を管理するDP**
  (例. dp[Left][Right] := 区間[Left, Right)の解)

N 匹のスライムが横一列に並んでいます。 最初、左から i 番目のスライムの大きさは a_i です。
太郎君は、すべてのスライムを合体させて 1 匹のスライムにしようとしています。
スライムが 1 匹になるまで、太郎君は次の操作を繰り返し行います。
左右に隣り合う 2 匹のスライムを選び、それらを合体させて新しい 1 匹のスライムにする。
合体前の 2 匹のスライムの大きさを
x および y とすると、合体後のスライムの大きさは x+y となる。
このとき、太郎君は x+y のコストを支払う。
なお、合体の前後でスライムたちの位置関係は変わらない。
太郎君が**支払うコストの総和の最小値**を求めてください。

$2 \leq N \leq 400$, $1 \leq a_i \leq 10^9$, 入力はすべて整数である。

- 方針と解答例
  - 逆に「スライムが分裂する際に、スライムの大きさだけコストがかかる」と考える
  - 分裂後の2つスライムの大きさが最小となる位置で分割する



Educational DP Contest / DP まとめコンテスト　N - Slimes https://atcoder.jp/contests/dp/tasks/dp_n

区間DP：コード

dp[l][r] := 区間[l, r)におけるコスト総和の最小値

https://atcoder.jp/contests/dp/submissions/4387250

```
// b[i] := 区間[0:i]の累積和
/*
    b[0] = a[0];
    for(int i=1; i<n; i++)
        b[i] = b[i-1]+a[i];
*/
//累積和から区間和取得
ll get(ll l, ll r){
    ll res = b[r];
    if(l) res -= b[l-1];
    return res;
}
```

```
ll dfs(ll l, ll r){
    if(l == r) return 0;
    if(visited[l][r]) return dp[l][r];
    visited[l][r] = true;
    ll res = INF;
    //分割をすべて試し, 最もコストが小さい位置で分割
    for(int i=l; i<r; i++) res = min(res, dfs(l, i)+dfs(i+1, r)+get(l, r));

    return dp[l][r] = res;
}
```

累積和を何も考えずに書けるようにする！ https://qiita.com/drken/items/56a6b68edef8fc605821

木DP (ツリーDP)

- 木構造上で計算するDP

N 個の島があります。 島には 1 から N までの番号がついています。 また、$N-1$ 個の橋があります。

i 番目の橋は a_i 番の島と b_i 番の島をつないでいます。

どの島からどの島へも橋をいくつか経由して到達できることがわかっています。
すぬけ君は、それぞれの島を白または黒に塗ることにしました。
ただし、両端の島が黒で塗られているような橋があってはいけません。

色の塗り方が何通りあるか、10^9+7で割った余りを求めてください。

$1 \leq N \leq 10^5$, $1 \leq a_i, b_i \leq N$, どの島からどの島へも橋をいくつか経由して到達できる

- 方針と解答例
  - 部分木の頂点 i における通り数を数える

dp[u][0] := 頂点 u を親とする部分木に
含まれる全頂点を白または黒で塗る通り数
dp[u][1] := 頂点 u が白の場合における上記

  - 下記参照

[漸化式を立てて「tree DP問題」を解く D - 塗り絵]

https://atcoder.jp/contests/abc036/submissions/7572854

```
static int dp[100001][2];
function< void(int,int) > dfs = [&](int u,int pre){
  dp[u][0] = dp[u][1] = 1;
  for(int v : G[u]){    //G[u][v] := 頂点uの子頂点 v
    if(v == pre) continue; //葉ならcontinue
    dfs(v,u); // 子頂点vのdp[v][*] を更新
    dp[u][0] = (dp[u][0] * (dp[v][0] + dp[v][1])) % MOD; //白+黒
    dp[u][1] = (dp[u][1] * dp[v][0]) % MOD;                //白のみ
  }
  return;
};
dfs(0,-1);
cout << (dp[0][0] + dp[0][1]) % MOD << endl;
```

AtCoder Beginner Contest 036 D - 塗り絵 https://atcoder.jp/contests/abc036/tasks/abc036_d

ゲームDP

- **左右の区間の状態を管理するDP**
  (例. dp[Left][Right] := 区間[Left, Right)の解)

N 個の正整数からなる集合 $A = a_1, a_2, \ldots, a_N$ があります。

太郎君と次郎君が次のゲームで勝負します。
最初に、K 個の石からなる山を用意します。 二人は次の操作を交互に行います。
先手は太郎君です。A の元 x をひとつ選び、山からちょうど x 個の石を取り去る。
先に操作を行えなくなった人が負けです。
二人が最適に行動すると仮定したとき、**どちらが勝つかを判定**してください。

入力はすべて整数である。$1 \leq N \leq 10^2$, $1 \leq K \leq 10^5$, $1 \leq a_1 < a_2 < \cdots < a_N \leq K$

- 方針と解答例
  - K = 0 の時 … 負け状態 (太郎の負け)
  - K = i の時 … K=0~i-1で負け状態が
    あるならば, 負けを相手に押し付ける

dp[i] := 残りの石が i 個における
太郎の勝ち状態フラグ

https://atcoder.jp/contests/dp/submissions/4365765

```
for(i=1; i<k+1; i++){ for(j=0; j<n; j++){
    //a[j]個の石を取れるなら, 取る
    //dp[i]=trueが1つでもあれば, true
    if(i-a[j]>=0){
      dp[i] |= !dp[i-a[j]];
    }
} }
```

Educational DP Contest / DP まとめコンテスト　K - Stones https://atcoder.jp/contests/dp/tasks/dp_k

部分列DP

- **部分文字列に関して計算するDP**
  (例. ある文字列において部分文字列の存在数を数え上げ)

長さ n の文字列 S が与えられる。
S の部分文字列 (空文字含む) として考えられるものの個数を数え上げよ。
ただし答えがとても大きくなることがあるので、**個数を 1,000,000,007 で割ったあまり**を求めよ。

$1 \leq n \leq 10^5$, S の各文字は小文字アルファベットのみ (26種類)

- 方針と解答例

dp[i] := 文字列 S のうち 0 から i-1番目の部分
より得られる部分文字列の個数

ただし, 同じ部分文字列を生成するなら選択 index が辞書順最小になるように

```
// next[i][c] := Sのi文字目以降で, 最初にcが登場するindex
vector<vector<int> > calcNext(const string &S) {
  int n = (int)S.size();
  vector<vector<int> > next(n+1, vector<int>(26, n));
  for (int i=n-1; i>=0; --i) {
    for (int j=0; j<26; ++j) next[i][j] = next[i+1][j];
    next[i][S[i]-'a'] = i;
  }
  return next;
}
```

(解答なし)

```
vector<vector<int> > next = calcNext(S);
dp[0] = 1;  //空文字も含む
for (int i=0; i<n; ++i) {
  for (int j=0; j<26; ++j) {
    if (next[i][j] >= n) continue;
    // dp[next[i][j]+1] に dp[i] を加算
    add(dp[next[i][j]+1], dp[i]);
  }
}
long long ans = 0;
for (int i=0; i<=n; ++i) add(ans, dp[i]);
cout << ans << endl;
```

Qiita 部分文字列の数え上げ 部分列 DP --- 文字列の部分文字列を重複なく走査する DP の特集

挿入DP

- 順列や文字列に挿入することで遷移するDP

以下の条件を満たす**文字列の個数を mod 1,000,000,007 で求めよ。**
a を $freq_1$ 個、b を $freq_2$ 個、... z を $freq_{26}$ 個含む (他の文字は含まない)。
同じ文字が隣り合うことはない。

$0 \leq freq_i \leq 10$, $freq_i$のうち, 少なくとも一つは0より大きいとする。

- 方針と解答例
  - 文字の間を見て両端が同文字ならng, 違う文字ならok
  - freq[i]の文字を分割して, ok/ngな箇所に入れていく

$$dp[i+1][j-l+freq[i]-k] += dp[i][j] * {}_{freq[i]-1}C_{k-1} * {}_{j}C_{l} * {}_{sum[i]+1-j}C_{k-l}$$

Algoogle　すいバカ日誌

dp[i][j] := i 文字目までにおいて, 連続している箇所 (=ngな箇所) が j 個あるような文字列の個数

分割数: k
現在のokな箇所の数 : sum[i]+1-j
次のngな箇所の数: j-l+freq[i]-k

```
#define repi(i,a,b) for(int i=(int)(a);i<(int)(b);i++)
#define rep(i,n) repi(i,0,n)
int solve(){
    //c := combination (場合の数), パスカルの三角形などで求める
    //f[i] := freq[i],  sum[i] := i-1番目までの総文字数
    rep(i,n) sum[i+1] = sum[i]+f[i];
    dp[0][0] = 1;
    rep(i,n) rep(j,sum[i]+1) repi(k,1,f[i]+1) rep(l,min(j,k)+1)
      add(dp[i+1][j-l+f[i]-k],
        mul(dp[i][j], mul(c[f[i]-1][k-1], mul(c[j][l], c[sum[i]+1-j][k-l]))));
    return dp[n][0];
}
```

Typical DP Contest　O - 文字列　https://tdpc.contest.atcoder.jp/tasks/tdpc_string

抜去DP

- 挿入DPとは逆に抜去するDP

それぞれ 1 から N の整数が 1 つずつ書かれた白いボールと黒いボールが合わせて $2N$ 個一列に並んでいます。
左から i $(1 \leq i \leq 2N)$ 個目のボールに書いてある数は a_i で、色は c_i で表されます。
$c_i = W$ のとき ボールが白いことを、$c_i = B$ のとき ボールが黒いことを表します。
人間の高橋君は次のような目標を達成したいです。
$1 \leq i < j \leq N$ を満たす任意の整数の組 (i, j) に対して、i が書かれた白いボールの方が j が書かれた白いボールより左にある
$1 \leq i < j \leq N$ を満たす任意の整数の組 (i, j) に対して、i が書かれた黒いボールの方が j が書かれた黒いボールより左にある
目標を達成するために高橋君は次のような操作ができます。
隣り合う二つのボールをスワップする目標を達成するために**必要な操作回数の最小値**を求めてください。

$1 \leq N \leq 2000$, $1 \leq a_i \leq N$, $c_i = W$ または $c_i = B$, $i \neq j$なら$(a_i, c_i) \neq (a_j, c_j)$

ARC097 E - Sorted and Sorted https://beta.atcoder.jp/contests/arc097/tasks/arc097_c

抜去DP：コード

- 方針と解答例
  - 白と黒について1から順番に先頭に取ってくる操作を行っていく
  - 重要考察「白のwまで、黒のbまで先頭に取ってきた場合に残るボールの列は一意に定まる」
    つまり、白と黒のどの数まで先頭に取ってきたかで状態を纏めることができそう
  - それをkeyとしたDPだとO(N^2)で、更新もO(1)で行けそう
- 下記参照

[はまやんはまやんはまやん]

dp[b][w] := 黒の数b, 白の数wまで先頭に持って来ている場合の操作回数の最小値

https://atcoder.jp/contests/arc097/submissions/2500398

```
int f(int b, int w) {
    if (b == N and w == N) return 0;
    if (0 <= dp[b][w]) return dp[b][w];
    int res = inf;
    if (b != N) {
        int cnt = 0;
        if (0 <= idxB[b + 1] - 1) cnt += smB[b + 1][idxB[b + 1] - 1];
        if (0 <= idxB[b + 1] - 1) cnt += smW[w + 1][idxB[b + 1] - 1];
        chmin(res, f(b + 1, w) + cnt);
    }
    if (w != N) {
        int cnt = 0;
        if (0 <= idxW[w + 1] - 1) cnt += smB[b + 1][idxW[w + 1] - 1];
        if (0 <= idxW[w + 1] - 1) cnt += smW[w + 1][idxW[w + 1] - 1];
        chmin(res, f(b, w + 1) + cnt);
    }
    return dp[b][w] = res;
}
```

ARC097　E - Sorted and Sorted　https://beta.atcoder.jp/contests/arc097/tasks/arc097_c

戻すDP

・ dp[i+1]からdp[i]を復元することで問題を解くDP

高橋商店では N 種類の商品が売られています。「どの種類の商品がいくつあるか」の情報が与えられるので、「合計 M 個の商品を選ぶ方法」の数を求めて下さい。ただし、同じ種類の商品は区別しないこととします。

いや、これは少し簡単過ぎるので、ちょっとした注文も追加しよう。整数 k, x からなる Q 個の注文を用意したので、それぞれについて「k_i種類目の商品をちょうど x_i 個選ばなければならないとき、**合計 M 個の商品を選ぶ方法」の数**を求めて下さい。

$1 \leqq N \leqq 2000$, $1 \leqq M \leqq 2000$, $1 \leqq Q \leqq 500{,}000$, $1 \leqq a_i \leqq M$, $1 \leqq k_i \leqq N$, $1 \leqq x_i \leqq a_{k_i}$

- 方針と解答例
  - 重複組合せを求める
  - 商品の見ていく順番は問わない点に着目して, 式変形
  - 変形により「N種類の商品を選ぶ選び方から1つ巻き戻す」ことができる

dp1[i][j] := i-1番目までにj個の商品を選ぶ通り数
dp2[i][j] := N種類の選び方から1つ巻き戻す用の配列

- 下記参照

kmjp's blog　ツバサの備忘録
ctyl's problem solving　mayoko’s diary

ARC028 D - 注文の多い高橋商店 https://arc028.contest.atcoder.jp/tasks/arc028_4

戻すDP：コード

dp1[i][j] := i-1番目まででj個の商品を選ぶ通り数
dp2[i][j] := N種類の選び方から1つ巻き戻す用の配列

```
FOR(i,N+1) dp1[i][0]=1;
//ここは通常通りdp1を埋める
FOR(i,N) FOR(j,M) {
    dp1[i+1][j+1]=dp1[i+1][j]+dp1[i][j+1];
    if(j-A[i]>=0) dp1[i+1][j+1]+=MOD-dp1[i][j-A[i]];
    dp1[i+1][j+1]%=MOD;
}
//前から逐一計算するのをやめて, 戻すDPで計算する
FOR(i,N) FOR(j,M+1) {
    dp2[i][j]=dp1[N][j];
    if(j) dp2[i][j]+=MOD-dp1[N][j-1];
    if(j-1-A[i]>=0) dp2[i][j]+=dp2[i][j-1-A[i]];
    dp2[i][j]%=MOD;
}
```

ARC028 D - 注文の多い高橋商店 https://arc028.contest.atcoder.jp/tasks/arc028_4

連結DP

- **連結関係に基づくDP**
  (例. dp[i][j] := i 列目までを使った時に, i列目のマス同士の連結関係がjであるような塗り方)
- Union findで連結する

すぬけ君は、$H \times W$ のマス目を白と黒に塗り分けることにした。
左上のマス目と右下のマス目は黒に塗る。
左上のマス目から、黒いマスを上下左右にたどって、右下のマス目に行くことができる。
条件を満たす塗り分け方の個数を mod 1,000,000,007 で求めよ。

$2 \leq H \leq 6$, $2 \leq W \leq 100$

- 方針と解答例
  - それぞれのマスの連結関係をもちながら, 列を左から右に見ていくDPを考える
  - 7進数bit(= 7^H)で以下のように考える:
    0 -> 白
    1 -> 黒 かつ (0, 0) と連結
    2~6 -> 黒 かつ (0, 0) と連結でない.
    同じ番号が振られていれば, それらのマスは連結であると考える.

dp[i][j] := i 列目まで見た時, i 列目の各マスの連結関係が j であるような塗り方

すいバカ日誌　kmjp's blog　simezi_tanの日記

```
for(int i = 0; i < w - 1; ++i) for(int j = 0; j < pw[h]; ++j) {  //次状態
  if(cur[j] == 0) continue;
  for(int k = 0; k < (1 << h); ++k) {
    union_find uf(h); vector<int> p(h); //uf: 連結関係を求める
    for(int l = 0; l < h; ++l) { //p: 次状態の各行の連結関係を保存
      if(k >> l & 1) {  // 次状態の l 行目が黒
        if(j / pw[l] % 7 == 1) p[l] = 1;  //現状態が連結なら次も連結
        for(int m = l + 1; m < h; ++m) { //連結関係をufでまとめる
          if(k >> m & 1) {
            if(m == l + 1) uf.unite(l, m);  //lとl+1行目が黒なら連結
            else {
              int u = j / pw[l] % 7,   //現状態のl行目, m行目の連結関係
                  v = j / pw[m] % 7;
              if(u > 0 && u == v) uf.unite(l, m); //現状態で同じ連結関係
            } …                                   //なら次状態でも連結
```

Typical DP Contest　S - マス目　https://tdpc.contest.atcoder.jp/tasks/tdpc_grid

Trie DP

・ Trie木の上で計算するDP

文字列 S と、要素数 M の単語の集合 P={P_1, P_2, …, P_M} が与えられます。単語 P_i は、整数の重み W_i を持っています。
文字列 S から、P に含まれる単語を重なり合わないように取り出すことを考えます。**単語の重みの総和が最大値をとるように取り出すとき、その最大値はいくつ**でしょうか？
なお、同じ単語を複数回取り出した場合、それらの単語は別々に数えることとします。

1≤|S|≤200000 , 1≤M≤5000 , 1≤|Pi|≤200 , 1≤Wi≤10000 , S, Pi は英小文字からなる文字列

- 方針と解答例
  - 与えられる単語すべてでTrie木をつくる
  - 文字列Sのi文字目s_iを左端とするSの部分文字列で, 単語と完全一致するものを考える
  - 単語の長さが高々200なので, 部分文字列の選び方は200通りしかない
- 下記参照
[hogecoder]

dp[i] := i 文字まで見た時の最大値

```
REP(i,N) {
  Trie* cur = &trie;
  repq(k,1,200) {
    if(i+k > N) continue;
    char target = s[i+k-1];
    Trie* next;
    if(cur != (Trie *)0) {
      next = trie.find(target, cur);  cur = next;
      chmax(dp[i+k], (cur != (Trie *)0) ? dp[i] + cur -> score : dp[i]);
    } else chmax(dp[i+k], dp[i]);
} }
```

AtCoder 天下一プログラマーコンテスト2016本戦 C - たんごたくさん
https://tenka1-2016-final-open.contest.atcoder.jp/tasks/tenka1_2016_final_c

全方位木DP

・任意の頂点を根とした木におけるDP

非負の重みをもつ**無向の木 T の直径を求めて**ください。木の最遠頂点間の距離を木の直径といいます。

$1 \leq n \leq 100{,}000$, $0 \leq w_i \leq 1{,}000$

- 方針と解答例
  - 適当な頂点を根として, 木を有向木とみなす
  - 任意の頂点を根とした部分木について, 部分木の根に “必要な情報D_i" を求める
    D_i : I番目の頂点から その頂点の部分木内で最も遠い頂点(葉) までの距離
  - 任意の頂点を根とした木全体について, 問題の解を求める

- 下記参照

ei1333の日記

```
void dfs1(int idx, int par){ for(edge &e : g[idx]) {
    if(e.to == par) continue;
    dfs1(e.to, idx);
    dist[idx] = max(dist[idx], dist[e.to] + e.cost);
}}
int dfs2(int idx, int d_par, int par){
  vector< pair< int, int > > d_child;
  d_child.emplace_back(0, -1);
  for(edge &e : g[idx]) {
    if(e.to == par) d_child.emplace_back(d_par + e.cost, e.to);
    else d_child.emplace_back(e.cost + dist[e.to], e.to);
  }
  sort(d_child.rbegin(), d_child.rend());
  int ret = d_child[0].first + d_child[1].first;
  for(edge &e : g[idx]) {
    if(e.to == par) continue;
    ret = max(ret, dfs2(e.to, d_child[d_child[0].second == e.to].first, idx));
  }
  return (ret);
}
dfs1(N / 2, -1);
cout << dfs2(N / 2, 0, -1) << endl;
```

Aizu Online Judge - Diameter of a Tree http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=GRL_5_A&lang=jp

インラインDP

- **更新箇所が少ない場合に，他の領域を使いまわすことで高速化するDP**
- **SegtreeやBITを用いて行う．**

N 本の花が横一列に並んでいます。 各 $i\ (1 \leq i \leq N)$ について、左から i 番目の花の高さは h_i で、美しさは a_i です。
ただし、$h_1, h_2, \ldots, h_N$ はすべて相異なります。

太郎君は何本かの花を抜き去ることで、次の条件が成り立つようにしようとしています。
残りの花を左から順に見ると、高さが単調増加になっている。
残りの花の美しさの総和の最大値を求めてください。

入力はすべて整数である。$1 \leq N \leq 2 \times 10^5$, $1 \leq h_i \leq N$, $h_1, h_2, \ldots, h_N$ はすべて相異なる。$1 \leq a_i \leq 10^9$

- 方針と解答例
  - dp[i] = max(dp[i], max(dp[0], dp[1], ..., dp[H[i-1]]) + A[i]というのをN回行う
  - dp[h]で区間maxのセグ木を作る
- 下記参照

はまやんはまやんはまやん

dp[h] := 右端の花の高さがhになるような並べ方の中における美しさの総和の最大値

```
rep(i, 0, N) {
  ll opt = dp.get(0, H[i]) + A[i];
  dp.update(H[i], max(dp[H[i]], opt));
}
cout << dp.get(0, 1<<18) << endl;
```

dp[i][j] = max{dp[i−1][k]|0≤k<j}+a[i] if h[i]==j , otherwise dp[i−1][j].
この時, dp[i]の更新は1箇所のみ (i-1 -> i の遷移が固定) なので, iを状態にもつ必要がない -> 添字を省略できる. <- 高速化!

Educational DP Contest / DP まとめコンテスト　Q - Flowers https://atcoder.jp/contests/dp/tasks/dp_q

Alien DP

・特定条件下でO(N^3)をO(N^2logN)に落とすDP

You are given an array A of N integers.
The cost of a subarray is defined as the bitwise or value of all its elements.
Suppose you split the array into KK non-empty subarrays and you compute the sum of their costs.
What is the maximum value you can get?

$1 \leq K \leq N \leq 2*10^5$, $0 \leq A_i < 2^{20}$

「f(j) = dp[i][j]」としたときに「f(j + 1) - f(j)≦f(j) - f(j - 1)」を満たしていれば、計算量を落とすことができる (?)

“何かをK回使ってコストを最小化してください，で，dp[pos][使った回数]だと遅すぎるときに，K回使う代わりに1回使うとx円罰金，にすればdp[pos]に出来る(こともある)テク(xで二分探索)”
by @yosupot

- コード

  https://csacademy.com/submission/1001327/

- 下記参照

  はまやんはまやんはまやん　Codeforces解説

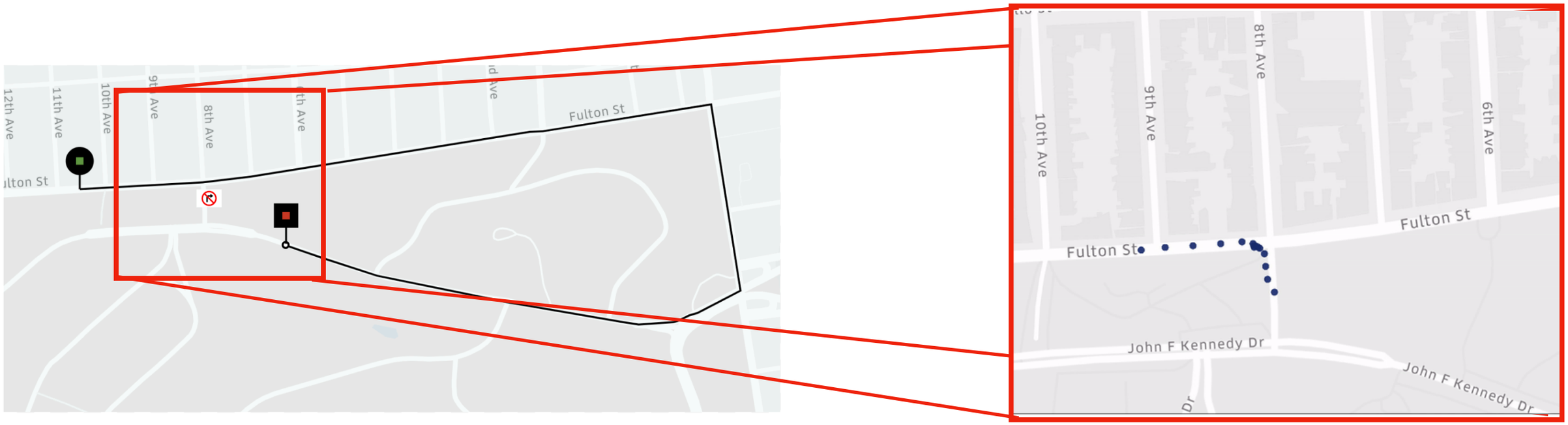
CSAcademy - Or Problem https://csacademy.com/contest/round-56/task/or-problem/

まとめ

- DPの種類は沢山あるよ！
- 頑張ってマスターしたい！！

参考: 配車GPSを活用した地図の自動更新 (Uber社)

- 隠れマルコフモデル(HMM)で車両をモデル化して[2] 車両の存在確率が高い位置を割り出し, **ビタビアルゴリズムと呼ばれるDPで, 車両が走行した尤もらしい経路を推定**する.



既存地図に基づくルート

車両走行データ

[2]: Improving Uber's Mapping Accuracy with CatchME
https://eng.uber.com/mapping-accuracy-with-catchme/