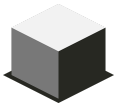
CCC beta #13: XMAS-XSAM

Editorial



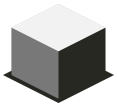
Santa's Gift by FM

สำหรับ 2 ข้อความ A กับ B นั้น longest uncommon subsequence ก็ควรจะเป็นไม่ A ก็ B เนื่องจาก ถ้าสมมุติให้ข้อความ C เป็น subseqeunce ของ A และเป็น uncommon subsequence ของ B แล้ว จะสรุปได้ว่า A จะไม่เป็น uncommon subsequence ของ B ด้วย ทำให้คำตอบเหลือเพียง A ไม่ก็ B ขึ้นอยู่ว่าอะไรยาวกว่ากัน

ยกเว้นกรณีที่ A = B ในกรณีนี้ถือว่าไม่มีคำตอบให้ตอบ -1

Solution Code :

```
#include <iostream>
#include <string>
#include <algorithm>

using namespace std;

int main() {
  string A, B;
  cin >> A >> B;
  if (A.compare(B) == 0)
        cout << -1;
  else
        cout << max(A.length(), B.length());
}
```



Christmas Tournament by Polpol

ข้อนี้ต้องทำ case-by-case analysis สมมุติให้ x, y, z เป็นจำนวนค้อน กรรไกร กระดาษ สามารถแบ่งกรณีได้ตามนี้

Case 1: มี ค้อน กรรไกร กระดาษ ทั้งสามอย่าง (x > 0, y > 0, z > 0) เราสามารถจัดแข่งตามลำดับนี้ได้

1.1 ให้ค้อนแข่งกับกระดาษทีละคน จนกระทั่งเหลือค้อนคนเดียว

1.2 ให้กระดาษแข่งกับกรรไกรจนกระดาษแพ้ออกไปหมด

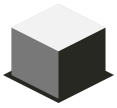
1.3 ให้กรรไกรทุกคนแข่งกันค้อนที่เหลือจาก 1.1 แล้วแพ้ออกจนหมดก็จะเหลือเพียงคนเดียว

Case 2: ไม่มีอย่างใดอย่างหนึ่ง ในกรณีนี้การแข่งจะเกิดขึ้นแบบเดียวก็คือ 2 อย่างที่เหลือต้องแข่งกัน ในกรณีจะมีผู้ชนะก็คือ คนชนะของทั้ง 2 แบบต้องมีคนเดียวตั้งแต่ต้นอยู่แล้วซึ่งมีได้ 3 แบบตาม solution code

Case 3: กรณีอื่นจะไม่สามารถทำให้มีผู้ชนะคนเดียวได้

Solution Code :

```
#include <iostream>
using namespace std;
int main() {
	int t;
	cin >> t;
	while(t--){
		int x,y,z;
		cin >> x >> y >> z;
		if(x>0 && y>0 && z>0){
			cout << "YES" << endl;
			continue;
		}
		if(x==1 && z==0){
			cout << "YES" << endl;
			continue;
		}
		if(y==1 && x==0){
			cout << "YES" << endl;
			continue;
		}
		if(z==1 && y==0){
			cout << "YES" << endl;
			continue;
		}
		cout << "NO" << endl;
	}
}
```

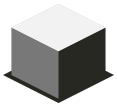


Santa Investor by PanTA

ข้อนี้เป็นปัญหา classic ที่แก้ได้โดยการใช้ greedy algorithm โดยตัว algorithm จะขายเมื่อราคานั้นดีกว่าราคาซื้อก่อนหน้า

Solution Code :

```
#include<stdio.h>

int main() {
	int n, x, y;
	y = 10000;
	int sum = 0;
	scanf("%d", &n);
	while (n--) {
		scanf("%d", &x);
		if (y <= x) {
			sum += x - y;
		}
		y = x;
	}
	printf("%d\n", sum);
	return 0;
}
```



The Night Before Christmas by phirasit

สำหรับข้อนี้เราจะเริ่มจากการลองแก้ไปทีละปัญหาย่อย ซึ่งเมื่อเรามองปัญหานี้เป็นกราฟก็คือเรามีกราฟเส้นตรง n โหนดที่เส้นบางเส้นถูกตัดออก โดยโจทย์จะให้ degree ของบางโหนดมาแล้วให้หากราฟเริ่มต้นมาให้ได้

ปัญหาย่อยที่ 1: ไม่มี ? ในข้อมูลนำเข้า

เราจะหาคำตอบจากการดู degree จากซ้ายไปขวาเนื่องจากต้นซ้ายสุดจะมีเส้นเชื่อมได้อย่างมากเส้นเดียว เราก็จะบอกว่ามีเส้นนี้หรือไม่ได้จาก degree อย่างเช่น 1... จะบอกได้ว่าคำตอบต้องเริ่มด้วย x- แน่ๆ แต่ถ้าข้อมูลนำเข้าเป็น 0... คำตอบจะเริ่มด้วย x. เราค่อยๆ ทำไปทีละตัวเรื่อยๆ ก็จะได้คำตอบทั้งหมด

ปัญหาย่อยที่ 2: มี ? เพียงหนึ่งตัว

ในส่วนนี้เรายังจะใช้วิธีแรกตราบใดที่ตัวทางซ้ายไม่ใช่ ? เมื่อเราพบ ? เราจะหาส่วนที่เหลือโดยการไล่จากขวาไปซ้ายแทนโดยวิธีการพิจารณาจะเหมือนกันแต่แค่เปลี่ยนทิศทาง เราจะได้กราฟทั้งหมดมา

ปัญหาย่อยที่ 3: คำตอบมีเพียงแบบเดียว

Observation ถ้า degree เป็น 2 เราจะบอกได้ว่ากราฟรอบจุดนั้นจะเป็น -x- และ ถ้า degree เป็น 0 เราจะบอกได้ว่ากราฟที่จุดนั้นจะเป็น .x. แน่ๆ

ถ้าเกิดเราแบ่งกราฟที่จุดเป็น 0 กับ 2 เป็นส่วนย่อยๆ เราจะรับประกันได้ว่าแต่ละส่วนจะมีเพียง 1 หรือ ? เท่านั้น ตัวอย่างเช่น 11111?1111 เพื่อรับประกันว่าคำตอบมีเพียงแบบเดียวตัว ? นั้นจะมีเพียงตัวเดียวเท่านั้นซึ่งเราสามารถใช้วิธีที่แก้ปัญหาย่อยที่ 2 แก้แต่ละส่วนย่อยได้

ปัญหาย่อยที่ 4: ไม่มี constraints ใด ๆ ให้ตอบคำตอบที่เป็นไปได้ 1 แบบ

ในส่วนนี้แต่ละปัญหาย่อยหลังจากแบบกราฟด้วยจุด degree 0 หรือ 2 แต่ละส่วนอาจจะมี ? มากกว่า 1 ตัวได้แต่เนื่องจากในส่วนย่อย ๆ นั้นไม่มี 0 กับ 2 แสดงว่าจะมี 1 เพียงอย่างเดียวเท่านั้น (ตัวอย่างที่เป็นไปได้เช่น ?11111?11?1?) เราก็สามารถเปลี่ยน ? เป็น 1 จนทำให้เหลือ ? เพียงตัวเดียวแล้วจึงใช้วิธีการแก้ตามปัญหาย่อยที่ 3 แก้ได้

Solution Code :

```
#include <cassert>
#include <iostream>

using namespace std;
```

```cpp
int main() {
  string str;
  cin >> str;

  const int n = str.length();

  // remove question marks
  // this part is not required if there is no ? in the input string
  bool noQbefore0or2 = true;
  for (int i = 0, parity = 0; i < n; ++i) {
    if (str[i] == '0' or str[i] == '2') {
      parity = (str[i] == '2');
      noQbefore0or2 = true;
    } else if (str[i] == '1') {
      parity = 1 - parity;
    } else if (str[i] == '?') {
      if (noQbefore0or2) {
        str[i] = (parity ? '#' : '*');
      }
      noQbefore0or2 = false;
    } else {
      assert(false); // invalid input
    }
  }
  for (int i = n-1, parity = 0; i >= 0; --i) {
    if (str[i] == '0' or str[i] == '2') {
      parity = (str[i] == '2');
    } else if (str[i] == '1') {
      parity = 1 - parity;
    } else if (str[i] == '?' || str[i] == '*') {
      str[i] = '0' + parity;
      parity = 0;
    } else if (str[i] == '#') {
      str[i] = '1' + parity;
      parity = 1;
    }
  }

  // cerr << "fixed string = " << str << endl;

  // generate output from a string of 0s, 1s, 2s
  cout << "x";
  for (int i = 1, parity = str[0] - '0'; i < n; ++i) {
    assert (0 <= parity and parity <= 1); // invalid string
    cout << (parity ? "-x" : ".x");
    parity = str[i] - '0' - parity;
    assert(0 <= parity and parity <= 1);
    assert(!(i == n-1 && parity != 0));
  }
  cout << endl;

  return 0;
}
```

กำหนดให้ $G=(V,E)$ แทนกราฟของปัญหานี้ $c: V \to \{`A', ..., `Z'\}$ แทนตัวอักษรในแต่ละโหนด และ $d: V \times V \to Z$ แทนระยะทางที่สั้นที่สุดระหว่างคู่โหนดใด ๆ บนกราฟ G

สำหรับในข้อนี้ เราสังเกตได้ว่าถ้าสตริงที่ให้มาเป็นคำว่า XMAS ไม่มี '_' ปรากฏอยู่ ณ ตำแหน่งใด ๆ เลย ปัญหานี้จะเป็นปัญหา Single-Source Shortest Path แบบทั่วไปที่สามารถใช้ Dijkstra's algorithm นำมาแก้ปัญหานี้ได้ แต่เมื่อมี '_' ปรากฏอยู่ในสตริงแล้ว เราจะไม่สามารถใช้ Dijkstra's algorithm ได้ตามปกติ จำเป็นต้องทำบางอย่างเพิ่มเติมเพื่อให้อัลกอริธึมทำงานได้ถูกต้อง

ในกรณีที่ '_' ปรากฏเพียงครั้งเดียวเท่านั้น ในกรณีนี้ เราสามารถใช้ Dijkstra's algorithm 2 ครั้ง ประกอบด้วยครั้งที่ให้จุดเริ่มต้นอยู่ที่โหนด 1 กับจุดเริ่มต้นอยู่ที่โหนด N ต่อจากนั้นทำการดูทุกโหนด v ที่มีตัวอักษรที่หายไปจากสตริง XMAS แล้วคำนวณหา $d(1,v)+d(v,N)$ มีค่าน้อยที่สุด

ส่วนในกรณีที่ '_' ปรากฏมากกว่าหนึ่งครั้ง เราจะสร้างกราฟใหม่ $G'=(V',E')$ ซึ่ง $V'=V \times P(\{`X', `M', `A', `S'\})$ และ $E'=\{((u,x),(v,y)) \in V' \times V' : (u,v) \in E \wedge y \subseteq x \cup \{c(v)\}\}$ จากนั้นเราใช้ Dijsktra's algorithm หาระยะทางที่สั้นที่สุดจาก $(1,s)$ ไปยัง $(N,\{`X', `M', `A', `S'\})$ ก็จะได้คำตอบที่ต้องการ เมื่อ s คือ เซตของตัวอักษรที่ปรากฏในสตริง XMAS

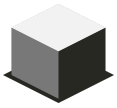
Solution Code :

```
#include <bits/stdc++.h>
#define N 50009
using namespace std;
typedef long long ll;
typedef pair<int,ll> pil;
typedef pair<ll,int> pli;

const ll inf = (1ll<<60);
const char word[7]="XMAS";

vector<pil> adj[N];
char c[N];
ll d[N][20];
bool vst[N][20];

int next(int state,char x){
	for(int i=0;i<4;i++){
		if(word[i]==x&&((1<<i)&state)==0)
			return state|(1<<i);
	}
	return state;
}

int decode(char *s){
	int ret=0;
	for(int i=0;i<4;i++){
		if(s[i]!='_') ret|=(1<<i);
```

```
	}
	return ret;
}

int main()
{
	int n,m; scanf("%d%d",&n,&m);
	char xmas[7]; scanf("%s",xmas);
	scanf("%s",c+2);
	while(m--){
		int u,v,w; scanf("%d%d%d",&u,&v,&w);
		adj[u].push_back({v,w});
		adj[v].push_back({u,w});
	}
	for(int i=1;i<=n;i++){
		for(int j=0;j<16;j++){
			d[i][j]=inf;
		}
	}

	priority_queue<pli> q;
	q.push({0,16+decode(xmas)});
	d[1][decode(xmas)]=0;
	while(!q.empty()){
		int u=q.top().second/16;
		int s=q.top().second%16;
		q.pop();
		if(vst[u][s]) continue;
		vst[u][s]=true;
		if(u==n&&s==15){
			printf("%lld\n",d[n][15]);
			return 0;
		}
		for(auto x: adj[u]){
			int v=x.first;
			int w=x.second;
			int t=next(s,c[v]);
			if(d[u][s]+w<d[v][t]){
				d[v][t]=d[u][s]+w;
				q.push({-d[v][t],16*v+t});
			}
		}
	}
	return 0;
}
```

CCC beta #14: Goodbye 2019

Editorial



Countdown 2020 by PanTA

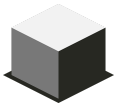
การทำ moving average เป็นปัญหา implementation ที่เขียนได้หลากหลายวิธี คือ

1. ทำ quick sum
2. ค่อยบวกเข้าและลบทิ้งไปเรื่อย ๆ

ในการแก้ข้อนี้สามารถใช้วิธีใดก็ได้ ใน solution code จะใช้วิธีที่สอง

Solution Code :

```
#include<stdio.h>
#include<queue>

int n, m;
std::queue<double> q;

int main() {
	scanf("%d %d", &n, &m);
	double sum = 0;
	double x;
	if (n < m) {
		printf("0\n");
	} else {
		printf("%d\n", n-m+1);
	}
	for (int i = 0; i < n; i++) {
		scanf("%lf", &x);
		q.push(x);
		sum += x;
		if (q.size() > m) {
			sum -= q.front();
			q.pop();
		}
		if (q.size() == m) {
			printf("%lf ", sum / (double)m);
		}
	}
	return 0;
}
```



New Year Decoration by PalmPTSJ

หมายเหตุ: การหาร ถ้าไม่ได้มีระบุเพิ่มเติม ให้ถือว่าเป็นการหารจำนวนเต็มบนภาษา C++ ซึ่งจะทำการหารแบบปัดเศษทิ้ง

Binary Search - O(log N)

วิธีทำข้อนี้ที่ง่ายที่สุดก็คือใช้ Binary Search โดยให้ binary search บนจำนวนเงิน (หรือจะใช้จำนวนกระดิ่งก็ได้) โดยถ้าลองเงิน X บาท จะสามารถคำนวณความสวยงามได้เป็น

beauty = X / 7 / 12 * 98 + X / 7 * 23

ถ้า beauty มีค่าน้อยกว่า N แสดงว่าคำตอบต้องมากกว่า X

ถ้า beauty มีค่ามากกว่าหรือเท่ากับ N แสดงว่าคำตอบต้องน้อยกว่าหรือเท่ากับ X

Better Solution - O(1)

เนื่องจากกระดิ่งทุก 12 ลูก จะได้ดาวแถมมา 1 ดวง เราสามารถมองได้ว่า เราซื้อกระดิ่งครั้งละ 12 ลูก เพื่อให้ได้ความสวยงาม 374 หน่วย (12 * 23 + 98) ได้ แล้วเราค่อยซื้อกระดิ่งเพิ่มเติมในส่วนที่ขาด ถ้ามองแบบนี้ เราจะต้องซื้อกระดิ่ง N / 374 * 12 ลูก แล้วเราจะต้องซื้อเติมอีก N % 374 หน่วย เพื่อความง่าย เรียกว่า X = N % 374

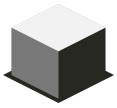
สำหรับส่วนที่ซื้อเติมนั้น ถ้ามองแบบเร็ว ๆ ก็คือใช้กระดิ่ง ceil(X / 23) ลูก แต่ว่าถ้าใช้สูตรนี้เลยจะผิด ถ้าให้ X = 373 จะเห็นว่า ceil(X / 23) = 17 แต่เราเห็นอยู่ว่าซื้อกระดิ่งแค่ 12 ลูก ก็ได้ 374 หน่วยแล้ว เหตุผลที่ผิดเพราะว่ากระดิ่งลูกที่ 12 นั้น เราจะได้ความสวยงาม 23 + 98 = 121 หน่วย ไม่ใช่แค่ 23 หน่วย เพราะจะได้ดาวแถมมาด้วย

ดังนั้น เราจะต้องดักไว้ด้วยว่า ถ้าเกิด X >= 276 ให้ตอบได้เลยว่าใช้กระดิ่งเพิ่มอีกแค่ 12 ลูก ถ้า X < 276 ถึงจะใช้สูตร ceil(X / 23) ได้ หรือไม่ก็ใช้ min(ceil(X / 23), 12) ได้เลย เพราะเรารับประกันได้ว่าไม่ว่า X จะเป็นเท่าไหร่ ก็ซื้อกระดิ่งไม่เกิน 12 ลูกแน่นอน

สรุปแล้ว ใช้กระดิ่ง N / 374 * 12 + min(ceil((N % 374) / 23), 12) ลูก

Solution Code :

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
int main() {
	ll n;
	scanf("%lld",&n);
	ll ans = n/374 * 12 + min((((n % 374) + 22) / 23), 12LL);
	printf("%lld", ans * 7);
}
```



New Year's Resolution by FM

โจทย์ข้อนี้เป็น dynamic programming โดยใน solution code จะมีตัวแปร dp[g][m] โดยจะมีค่าเท่ากับเงินที่น้อยที่สุดที่ทำให้คุณมีแฟน g คนในเดือนที่ m โดย transition ที่เป็นไปได้คือ

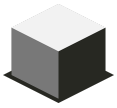
dp[i][j] = min(dp[i][j], dp[k][j - 1] + (i - k) * x + i * y) ในเดือนที่ j มีการจ้างแฟนขึ้นไปเป็น i คน

dp[i][j] = min(dp[i][j], dp[k][j - 1] + (k - i) * z + i * y) ในเดือนที่ j จะมีการเลิกกับแฟนให้เหลือ i คน

คำตอบก็คือ min (dp[i][n]) ที่ i สูงกว่า minimum แฟนของเดือนสุดท้าย

Solution Code :

```
#include <iostream>
#include <algorithm>
using namespace std;
int dp[55][1010];
void init() {
  for (int i = 0; i <= 50; i++)
        for (int j = 0; j <= 1000; j++)
          dp[i][j] = -1;
}
int main() {
  init();
  int n;
  cin >> n;
  int x, y, z;
  cin >> x >> y >> z;
  dp[0][0] = 0;
  for (int j = 1; j <= n; j++) {
        int minimum;
        cin >> minimum;
        for (int i = minimum; i <= 50; i++) {
          dp[i][j] = 2e9;
          for (int k = 0; k <= 50; k++) {
                if (dp[k][j - 1] != -1) {
                  if (k <= i)
                        dp[i][j] = min(dp[i][j], dp[k][j - 1] + (i - k) * x + i * y);
                  else
                        dp[i][j] = min(dp[i][j], dp[k][j - 1] + (k - i) * z + i * y);
                }
          }
        }
  }
  int answer = 2e9;
  for (int i = 0; i <= 50; i++){
        if (dp[i][n] != -1)
          answer = min(answer, dp[i][n]);
  }
  cout << answer;
}
```



Marineford Boxing Day by Minato

พิจารณากองของขวัญ ขนาดกว้าง A กล่อง ยาว B กล่อง สูง C กล่อง วางลงในระบบพิกัดฉาก 3 มิติ
การยิงลำแสงจากมุมนึงไปยังมุมนึงไปยังมุมตรงข้าม คือการลากเส้นตรงจากจุด (0, 0, 0) ไปยังจุด (A, B, C) เรียกว่าเส้น D
ให้ G(x, y, z) หมายถึง กล่องของขวัญกล่องที่ x ตามแนวแกน x, กล่องที่ y ตามแนวแกน y และกล่องที่ z ตามแนวแกน z
โดยที่ $0 \le x \le A - 1$, $0 \le y \le B - 1$, $0 \le z \le C - 1$

พิจารณา เส้น D พบว่า เมื่อเราเริ่มเดินตามเส้นจากจุด (0, 0, 0) ไปยังจุด (A, B, C)
เห็นได้ชัดว่า กล่องแรกที่โดนคือ G(0, 0, 0) และกล่องสุดท้ายที่โดนคือ G(A - 1, B - 1, C - 1)
และเมื่อลากมาถึง G(i, j, k) แล้ว กล่องถัดไปที่เส้น D จะโดนมี 3 กรณี

<u>กรณีที่ 1</u> G(i + 1, j, k), G(i, j + 1, k) หรือ G(i, j, k + 1)
เมื่อเส้น D วิ่งออกจาก G(i, j, k) ทางด้านที่ตั้งฉากกับแกน x, y หรือ z ตามลำดับ
เมื่อไล่ G(i, j, k) จาก G(0, 0, 0) ไปยัง G(A - 1, B - 1, C - 1) พบว่า ค่า i, j และ k จะมีการเพิ่มขึ้น A - 1, B - 1 และ C - 1 ครั้ง ตามลำดับ
หรือก็คือซึ่งเหตุการณ์นี้เป็นจำนวน A - 1, B - 1 และ C - 1 ครั้ง ตามลำดับ
ให้เป็น F(1, 0, 0), F(0, 1, 0) หรือ F(0, 0, 1) ตามลำดับ

<u>กรณีที่ 2</u> G(i, j + 1, k + 1), G(i + 1, j, k + 1) หรือ G(i + 1, j + 1, k)
เมื่อเส้น D วิ่งออกจาก G(i, j, k) ทางเส้นขอบที่ขนานกับแกน x, y หรือ z ตามลำดับ
ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อ j + 1 : k + 1 = B : C, i + 1 : k + 1 = A : C หรือ i + 1 : j + 1 = A : B ตามลำดับ
เป็นจำนวน GCD(B, C) - 1, GCD(A, C) - 1 หรือ GCD(A, B) - 1 ครั้ง ตามลำดับ
ให้เป็น F(0, 1, 1), F(1, 0, 1) หรือ F(1, 1, 0) ตามลำดับ

<u>กรณีที่ 3</u> G(i + 1, j + 1, k + 1)
เมื่อเส้น D วิ่งออกจาก G(i, j, k) ทางมุมพิกัด (i + 1, j + 1, k + 1) พอดี
ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อ i + 1 : j + 1 : k + 1 = A : B : C
เป็นจำนวน GCD(A, B, C) - 1 ครั้ง
ให้เป็น F(1, 1, 1)

ซึ่งทุกครั้งที่เรานับ G(i + 1, j, k) และ G(i, j + 1, k) จะมีบางครั้งที่เรานับซ้ำกันกับ G(i + 1, j + 1, k) และในทำนองเดียวกันกับกรณีอื่น ๆ

เราจึงใช้หลักการนำเข้าตัดออก (PIE)

ได้ F(1, 0, 0) + F(0, 1, 0) + F(0, 0, 1) - F(0, 1, 1) - F(1, 0, 1) - F(1, 1, 0)

= (A - 1) + (B - 1) + (C - 1) - (GCD(B, C) - 1) - (GCD(A, C) - 1) - (GCD(B, C) - 1) + (GCD(A, B, C) - 1)

= A + B + C - GCD(A, B) - GCD(A, C) - GCD(B, C) + GCD(A, B, C) - 1

นั่นคือเมื่อไล่จาก G(0, 0, 0) ถัดไป A + B + C - GCD(A, B) - GCD(A, C) - GCD(B, C) + GCD(A, B, C) - 1 กล่อง จะพบ G(A - 1, B - 1, C - 1) ซึ่งเป็นกล่องสุดท้าย

เพราะฉะนั้นเมื่อรวม G(0, 0, 0) เข้าไปด้วย

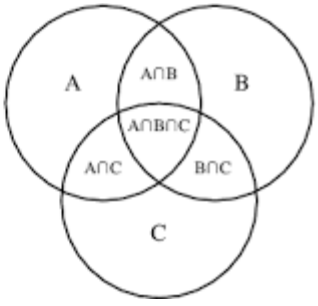
จึงมีกล่องที่เสียหาย A + B + C - GCD(A, B) - GCD(A, C) - GCD(B, C) + GCD(A, B, C) กล่อง

Solution ที่สอง

ในการนับว่าส่วนของเส้นตรง (0, 0, 0) ไปยัง (A, B, C) นั้นผ่านทรงลูกบาศก์จำนวนกี่ลูก สามารถทำได้โดยการนับว่าเส้นนั้นผ่านขอบลูกบาศก์จำนวนกี่ครั้งแทนได้ (ในที่นี้ไม่นับ (0, 0, 0) ว่าเป็นขอบกล่อง)

นิยามของจุดบนเส้นระหว่าง (0, 0, 0) กับ (A, B, C) ทุกจุดบนเส้นสามารถเขียนอยู่ในรูป α(A, B, C) โดยที่ $0 \le \alpha \le 1$

นิยามขอบกล่อง ขอบกล่องคือจุด (x, y, z) ที่ x, y หรือ z มีค่าเป็นจำนวนเต็ม ด้วยนิยามของจุดบนเส้นและขอบกล่องทำให้ปัญหาก็จะกลายเป็นจำนวน α ที่ทำให้ αA หรือ αB หรือ αC เป็นจำนวนเต็ม



โดยการนับเราสามารถทำได้โดยใช้หลักการ Principle of Inclusion and Exclusion นั้นจะได้คำตอบคือ

G(A) + G(B) + G(C) - G(A, B) - G(B, C) - G(A, B) + G(A, B, C)

ลองนับจาก Venn Diagram ทางขวาก็ได้

G(x) คือจำนวน α ระหว่าง 0 กับ 1 ที่ทำให้ αx เป็นจำนวนเต็ม ซึ่ง G(x) = x เพราะว่า ทุกๆค่า $1 \le k \le x$ เราสามารถให้ค่า $0 \le (\alpha = k/x) \le 1$ ได้

G(x, y) คือจำนวน α ที่ทำให้ αx และ αy เป็นจำนวนเต็ม ค่า G(x, y) = GCD(A, B) เนื่องจากถ้า αx, αy เป็นจำนวนเต็มแล้ว αGCD(x, y) ต้องเป็นจำนวนเต็มด้วย (เพราะ GCD(x, y) สามารถเขียนในรูป ax + by โดย a, b เป็นจำนวนเต็มได้) ทำให้ α ทุกตัวมีค่าได้เพียง k/GCD(x, y) โดยที่ $1 \le k \le$ GCD(x, y) เท่านั้น แต่ทุก α ในนี้จะทำให้ αx และ αy เป็นจำนวนเต็มเพราะ kx/GCD(x,y) และ ky/GCD(x,y) นั้นเป็นจำนวนเต็มเสมอ ทำคำตอบเป็นไปได้ GCD(x, y)

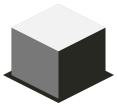
แบบเท่านั้น

G(x, y, z) = GCD(x, y, z) ด้วยเหตุผลคล้าย ๆ กัน

ทำให้คำตอบของข้อนี้เท่ากับ A + B + C - GCD(A, B) - GCD(B, C) - GCD(A, C) + GCD(A, B, C)

Solution Code :

```c
#include <stdio.h>

long long GCD(long long x, long long y){
    while((x != 0) && (y != 0)){
        if(x > y) x = x % y;
        else y = y % x;
    }
    return x + y;
}

int main()
{
    long long a, b, c;
    scanf("%lld %lld %lld", &a, &b, &c);
    printf("%lld", a + b + c - GCD(a, b) - GCD(a, c) - GCD(b, c) + GCD(GCD(a, b), c));
    return 0;
}
```



New Year Gift Cost by PeaTT

เราจะมองการเลือกมูลค่าของของขวัญเป็นการค่อย ๆ วางเลขโดดที่เรามีจากหลักที่มีค่ามากไปหลักที่มีค่าน้อยแทน

เมื่อเราอยากให้มูลค่าของขวัญน้อยที่สุด เราก็อยากจะให้เลขหลักทางซ้ายของมูลค่ามีค่าน้อยที่สุดโดยที่ยังสามารถวางเลขที่เหลือได้อยู่โดยไม่ขัดกับเลขต้องห้าม

ลองพิจารณา Strategy ในการวางเลขต่อไปนี้

1. ถ้ามีเลขโดดที่มีจำนวนที่ว่าง (ที่ ๆ เลขต้องห้ามไม่เป็นเลขนั้น) พอให้วาง “พอดี” ให้เราวางเลขนั้น (ถ้ามีหลายเลข จะวางเลขไหนก่อนก็ได้) ลงบนช่องว่างให้หมด แล้วช่องเลขต้องห้ามที่เหลือจะเป็นของเลขโดดเลขนั้น ดังนั้น เราสามารถวางเลขที่เหลือของทั้งหมดที่เหลือลงได้เลย

2. ถ้าไม่มีเลขโดดดังกล่าว ให้เราวางเลขไปเรื่อย ๆ โดยไม่ขัดกับเลขต้องห้ามจนกว่าจะเกิดเหตุการณ์ในข้อ 1

จาก strategy ข้างต้น ทำให้รู้ว่า ตราบเท่าที่เลขโดดทุกเลขยังมีที่ว่างพอให้ตัวเองวาง เราก็จะสามารถวางเลขโดดทั้งหมดได้เสมอ ทำให้เกิด greedy strategy ดังต่อไปนี้

1. ให้วางเลขจากหลักมากไปน้อย
2. พิจารณาว่าตอนนี้มีเลขโดดที่มีช่องเหลือว่างเหลือ “พอดี” อยู่หรือไม่
   - ถ้าไม่มี ให้เรา greedy วางเลขโดดที่น้อยที่สุดที่ยังเหลืออยู่ที่ไม่ตรงกับเลขต้องห้ามลงไปเพราะยังไงที่เหลือก็วางได้เสมอ
   - แต่ถ้ามี ให้ดูว่าหลักที่เราจะวางมีเลขต้องห้ามตรงกับเลขที่เราจะวางหรือไม่ เพราะถ้าตรงเรายังไม่จำเป็นต้องวางเลขโดดนั้นที่นี่ แล้วก็ greedy ตามปกติ แต่ถ้าไม่ตรง เราต้องวางเลขนั้นที่นี่ไม่งั้นจะทำให้ที่ว่างของเลขโดดนี้เหลือไม่พอ ทำให้ไม่สามารถวางเลขที่เหลือได้ ถ้ามีหลายเลข (เช่น กรณีมีเลขต้องห้ามเป็น 11112222 แล้วเรามี 1 และ 2 อย่างละสี่ตัว) ให้เราวางอันที่วางได้ลงไป
3. วางไปเรื่อย ๆ จนกว่าเลขโดดของเราจะหมด

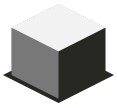
Time Complexity & Space Complexity : O(E)

Solution : Greedy Algorithm

Solution Code :

```cpp
#include <cstdio>

int check(int nums[5], int forbid[5]) {
   int all = nums[1]+nums[2]+nums[3]+nums[4];
   for (int i=1; i<=4; i++) {
      if (all-nums[i] < forbid[i]) return 0;
   }
```

```
    return 1;
}

int nums[5], forbidCnt[5], forbid[100010], n, found = 0;
char tmp[100010], ans[100010];

void solve() {
    int f;
    for (int state=0; state<n; state++) {
        f = forbid[state];
        forbidCnt[f]--;
        for (int i=1; i<=4; i++) {
            if (i == f || nums[i] == 0) continue;
            nums[i]--;
            ans[state] = '0' + i;
            if (check(nums, forbidCnt)) break;
            nums[i]++;
        }
    }
    ans[n] = 0;
    printf("%s\n", ans);
}


int main () {
    int q;
    scanf("%d", &q);
    while (q--) {
        scanf("%d%d%d%d%d", &nums[1], &nums[2], &nums[3], &nums[4], &n);
        scanf(" %s", tmp);
        for (int i=0; i<n; i++){
            forbid[i] = tmp[i] - '0';
            forbidCnt[tmp[i] - '0']++;
        }
        solve();
    }
    return 0;
}
```



Dice Force by Minato

ข้อนี้เป็น DP โดยมองเป็นปัญหาการหยิบของใส่ถุงให้ได้มูลค่ามากที่สุดโดยที่น้ำหนักไม่เกินที่ถุงจะรับไหว
แต่ละหน้าของลูกเต๋าที่เราสร้าง ก็คือน้ำหนักของสิ่งของที่เราจะหยิบ
และเมื่อนำไปเปรียบเทียบกับทุกหน้าของลูกเต๋าของเรนแล้ว ถ้าชนะให้มูลค่า +1 ถ้าเสมอให้เท่าเดิม ถ้าแพ้ให้ -1
ในกรณีที่สิ่งของมูลค่าเท่ากัน ให้สนใจแค่อันที่มีน้ำหนักน้อยกว่า

ตัวอย่างเช่น ลูกเต๋าของเรนมี 6 หน้า เป็น 1, 1, 2, 3, 5, 8 เมื่อนำไปแปลงเป็นสิ่งของ เราจะได้

ชิ้นที่	0	1	2	3	4	5	6	7	8
น้ำหนัก (W)	0	1	2	3	4	5	6	8	9
มูลค่า (V)	-6	-4	-1	1	2	3	4	5	6

ส่วนน้ำหนักอื่น ๆ ไม่ต้องสนใจ เพราะเรามีของที่มูลค่าเท่ากัน แต่มีน้ำหนักน้อยกว่าทดแทนได้อยู่แล้ว

ให้ W[i] คือน้ำหนักของของชิ้นที่ i, V[i] คือมูลค่าของของชิ้นที่ i
ให้ DP[x][y] หมายถึงมีของ x ชิ้น น้ำหนัก y พอดีที่มีมูลค่ามากที่สุดที่เป็นไปได้
ได้ DP[x][0] = x * V[i]
และ DP[x][y] = MAX(DP[x - 1][y - W[i]] + V[i]) สำหรับทุก i ที่ y - W[i] >= 0

แต่ลูกเต๋าของลุคต้องมี N หน้า และผลรวมของแต่ละหน้าเป็น S พอดี
นั่นคือต้องมีของ N ชิ้นพอดี น้ำหนัก S พอดี
แต่จากของที่เรานำมาคำนวณ เราตัดของที่มีมูลค่าเท่ากันแต่น้ำหนักมากกว่าออก
ทำให้ในกรณีที่มีของ N ชิ้น แต่น้ำหนักน้อยกว่า S เราสามารถหาน้ำหนักมาชดเชยจนเป็น S ได้ โดยที่มีมูลค่าเท่าเดิม

เพราะฉะนั้นคำตอบของข้อนี้คือ $\max_{0 \le j \le S} DP[N][j]$

และเมื่อได้ค่ามาแล้ว ให้ไล่ย้อนกลับไปว่ามาจากสิ่งของน้ำหนักเท่าไหร่บ้าง
ในกรณีที่น้ำหนักรวมน้อยกว่า S ก็บวกน้ำหนักเข้าไปที่ของชิ้นใดก็ได้จนเท่ากับ S

Solution Code :

```
#include <bits/stdc++.h>
#define N 60
#define M 5009
#define inf 10000009
using namespace std;
int dp[N][M];
int tr[N][M];
int dice[N],ans[N];
int main() {
    int n,k; scanf("%d%d",&n,&k);
    int s=0;
    for(int i=0;i<n;i++){
        scanf("%d",dice+i);
        s+=dice[i];
    }
    sort(dice,dice+n);
    vector<int> w,v;
    w.push_back(0);
    for(int i=0;i<n;i++){
        if(w.back()<dice[i]) w.push_back(dice[i]);
        if(w.back()<=dice[i]) w.push_back(dice[i]+1);
    }
    int p=0;
    for(int i=0;i<w.size();i++){
        while(p<n&&dice[p]<w[i]) p++;
        int q=p;
        while(q<n&&dice[q]<=w[i]) q++;
        v.push_back(p+q-n);
        p=q;
    }
    for(int k=0;k<=s;k++) dp[0][k]=-inf;
    dp[0][0]=0;
    tr[0][0]=-1;
    p=0;
    for(int i=1;i<=n;i++){
        for(int k=0;k<=s;k++){
            dp[i][k]=-inf;
            for(int j=0;j<w.size();j++){
                if(k>=w[j]&&dp[i][k]<dp[i-1][k-w[j]]+v[j]){
                    dp[i][k]=dp[i-1][k-w[j]]+v[j];
                    tr[i][k]=j;
                }
            }
            if(i==n&&dp[n][k]>dp[n][p]) p=k;
        }
    }
    int r=s-p;
    for(int i=n;i>0;i--){
        int t=w[tr[i][p]];
        ans[i-1]=t;
        p-=t;

        if(r>0){
            int tt=*upper_bound(dice,dice+n,t);
            if(tt<r) ans[i-1]+=tt, r-=tt;
            else ans[i-1]+=r, r=0;
        }
    }
    for(int i=0;i<n;i++)
        printf("%d ",ans[i]);
    return 0;
}
```