tugrader Contest 1 Editorial

Problem List

Caesar Cipher

Da Vinci Code

Supernova 2

Triangle Art

Shortest Path on a Ternary Tree

String

Caesar Cipher

การเข้ารหัสแบบซีซาร์ เป็นการเข้ารหัสอย่างง่ายแบบหนึ่ง โดยจะเป็นการเลื่อนตัวอักษรไปเรื่อยๆทีละตัว เราสามารถ loop ไล่ไปทีละตัวอักษรแล้วเลื่อนโดยแปลงตัวอักษรเป็นตัวเลขในระบบแอสกี (ASCII) จากนั้นก็นำไปบวกกับค่า k เพื่อเลื่อน แล้วก็นำมาหาเศษจากการหารด้วย 26 ซึ่งถือว่าเป็นคาบของการเลื่อนตัวอักษร (ทุกๆ 26 ตัวจะกลับมาที่เดิม)

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
char s[1048576];
int main(){
  int n,k;
  scanf("%d%d",&n,&k);
  scanf("%s",s);
  for(int i = 0; i < n; i++){
    if(isupper(s[i])){
      int buf = (s[i] - 'A' + k) % 26;
      if(buf < 0) buf += 26;
      s[i] = buf + 'A';
    }else{
      int buf = (s[i] - 'a' + k) % 26;
      if(buf < 0) buf += 26;
      s[i] = buf + 'a';
    }
  }
  printf("%s",s);
}
```

Da Vinci Code

โจทย์ข้อนี้เป็นการใช้กำหนดการพลวัต (Dynamic Programming) เพื่อเพิ่มประสิทธิภาพการทำงานของโปรแกรม สังเกตว่าตัวเลขที่ให้ไปนั้นยาวมาก จึงเก็บในรูปแบบสตริง โดยเราจะคิดความสัมพันธ์เวียนเกิดสำหรับข้อนี้

ให้ A_i แทนคำตอบ เมื่อพิจารณาสตริงในตำแหน่งที่ 1 ถึง i จะได้ความสัมพันธ์ของ A_i เป็นดังนี้

$$A_i = A_{i-2} * \Delta_{i-1,i} + A_{i-1} * \Delta_i$$

โดยที่ Δ เป็น Operator ที่จะให้ผลลัพธ์ 0 เมื่อสตริงไม่สามารถเกิดได้ แต่จะให้ผลลัพธ์ 1 เมื่อสามารถเกิดสตริงได้ (ในกรณี $\Delta_{i-1,i}$ หากจำนวนที่ได้จากการพิจารณาสตริงตัวที่ i-1 และ i มีค่า อยู่ระหว่าง 1 ถึง 26 และ i-1 ไม่เท่ากับ 0 จะถือว่ามีโอกาสเป็นไปได้ ส่วน Δ_i จะได้จากการพิจารณาว่า สตริงตัวที่ i เมื่อมองเป็นตัวเลขจะมีค่าอยู่ระหว่าง 1 ถึง 9 หรือไม่

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
const int MOD = 85142019;
char s[1048576];
int dp[1048576];
int d(int x){
     return 1 <= x && x <= 26 ? 1 : 0;
}
int main(){
     scanf("%s",s+1);
     int len = strlen(s+1);
     dp[0] = 1;
     dp[1] = d(s[1]-48);
     for(int i = 2; i <= len; i++){
          dp[i] = (
(long long)dp[i-1] * d(s[i]-48) +
(long long)dp[i-2] * d(s[i-1] == '0' ? 0 : s[i-1]*10 - 480 + s[i]
- 48)
) % (long long)MOD;
     }
     printf("%d",dp[len]);
}
```

Supernova II

ในข้อนี้เราจะมองว่าประเทศคือกราฟ ส่วนเมืองคือบางส่วนของกราฟ โดยหากพิจารณากราฟแล้วจะมีลักษณะมีจำนวนจุดยอดเท่ากับจำนวนเส้นเชื่อม ก่อนอื่นเราต้องการหา cycle บนกราฟ ซึ่งสามารถใช้ Depth First Search เพื่อหา cycle ได้ และสามารถระบุได้ด้วยว่าจุดยอดใดอยู่ภายใน cycle

เมื่อสามารถหา cycle ได้แล้ว ก็ค่อยๆตัดทุกเส้นบน cycle ออก (สังเกตว่าแต่ละจุดบน cycle คือบ้านของผู้ปกครองเมือง แต่ละเมือง) โดยในความเป็นจริงเราไม่จำเป็นต้องตัดออกก็ได้ แค่สมมติว่ามันไม่มี

หลังจากนั้นสังเกตว่ากราฟจะกลายเป็น forest แล้ว สิ่งที่เราต้องหาคือ ขนาดของ forest ที่ใหญ่ที่สุดลบออกด้วย ขนาดของ forest ที่เล็กที่สุด ซึ่งสามารถหาได้โดยการ Depth First Search บนแต่ละ Tree ใน Forest

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
const int MXN = 5e5+5;
vector<int> g[MXN];
vector<int> cy;
bool ic[MXN];
int p[MXN];
bool dfs(int c,int pp = -1){
	if(p[c]) return false;
	p[c] = pp;
	for(int i = 0; i < g[c].size(); i++){
		if(g[c][i] == pp) continue;
		if(p[g[c][i]]){
			int x = c;
			while(x != g[c][i]){
				cy.push_back(x);
				ic[x] = true;
				x = p[x];
			}
			cy.push_back(g[c][i]);
			ic[g[c][i]] = true;
			return true;
		}
		if(dfs(g[c][i],c)) return true;
	}
	return false;
}
int dfsc(int x,int p = -1){
	int s = 1;
	for(int y : g[x]){
```

```
            if(y == p || ic[y]) continue;
            else s += dfsc(y,x);
      }
      return s;
}
int main(){
      int n;
      scanf("%d",&n);
      int x;
      for(int i = 1; i <= n; i++){
            scanf("%d",&x);
            g[i].push_back(x);
            g[x].push_back(i);
      }
      dfs(1);
      int c;
      int mx,mn;
      mx = 0;
      mn = 1e9;
      for(int x : cy){
            c = dfsc(x);
            mx = max(mx,c);
            mn = min(mn,c);
      }
      printf("%d",mx-mn);
      return 0;
}
```

Triangle Art

Problem Credit: https://practice.thailandoi.org/

ข้อนี้เป็นการนับจำนวนรูปแบบของการสร้างศิลปะสามเหลี่ยม ซึ่งแก้ได้ด้วยกำหนดการพลวัต เราพิจารณาความสัมพันธ์เวียนบังเกิด $R_{i,j}$ = จำนวนวิธีการสร้างศิลปะสามเหลี่ยมเมื่อแถวแรกยาว i แถวที่สองยาว j จะได้ความสัมพันธ์ดังนี้

$$R_{i,j} = R_{i-1,j} + R_{i,j-1} - R_{i-1,j-1} + \sum_{l=\max(i-k,1)}^{i-1} (1 + R_{l-1,j-1}) + \sum_{l=\max(j-k,1)}^{j-1} (R_{l-1,j-1})$$

ความสัมพันธ์ดังกล่าวพิจารณาจากกรณีที่ไม่มีจุดด้านบนและไม่มีจุดด้านล่าง จะได้ $R_{i-1,j} + R_{i,j-1} - R_{i-1,j-1}$ (จากหลักการเพิ่มเข้าตัดออก) และอีกกรณีคือใช้ i,j เป็นด้านหนึ่งของสามเหลี่ยมปัจจุบัน จะได้ว่ามีได้หลายแบบ โดยพิจารณาจากการวางสามเหลี่ยมทั้งหมดจึงได้เป็น $\sum_{l=\max(i-k,1)}^{i-1}(1 + R_{l-1,j-1}) + \sum_{l=\max(j-k,1)}^{j-1}(R_{l-1,j-1})$

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
const int MXN = 128;
int dp[MXN][MXN];
int main(){
	int n,k;
	scanf("%d%d",&n,&k);
	for(int i = 1; i <= n; i++){
		for(int j = 1; j <= n; j++){
			dp[i][j] = dp[i-1][j] + dp[i][j-1] - dp[i-1][j-1];
			dp[i][j] %= 1000000007;
			for(int l = max(j-k,1); l < j; l++){
				dp[i][j] += dp[i-1][l-1]+1;
				dp[i][j] %= 1000000007;
			}
			for(int l = max(i-k,1); l < i; l++){
				dp[i][j] += dp[l-1][j-1]+1;
				dp[i][j] %= 1000000007;
			}
		}
	}
	printf("%d",dp[n][n]);
}
```

Shortest Path on a Ternary Tree

จากโจทย์สังเกตได้ว่า หากเราอยู่ที่จุดยอดหมายเลข i สามารถไปหาจุดยอดพ่อแม่ได้ โดยจุดยอดพ่อแม่จะมีค่าเท่ากับ $\left\lfloor \frac{i+1}{3} \right\rfloor$ ดังนั้นเราจึงนำเลขแสดงจุดยอดปัจจุบันมาบวกกับหนึ่ง แล้วหารด้วยสาม แต่เนื่องจากเป็นเลขฐานสามจึงสามารถบวกด้วยหนึ่งและหารด้วยสามได้ง่าย การบวกต้องบวกแบบปกติ หากมีตัวทดต้องทดไปเรื่อยๆ และการหารด้วยสาม ทำได้โดยการละทิ้งหลักสุดท้ายของเลขฐานสามไป

เราสามารถหาเส้นทางที่สั้นที่สุดของจุดยอด x และ y ได้โดยการค่อยๆหาจุดยอดพ่อแม่ของ x ไปเรื่อยๆจนถึงรากแล้วบันทึกแต่ละอันไว้ หลังจากนั้นก็ค่อยๆหาจุดยอดพ่อแม่ของ y หากจุดยอดปัจจุบันอยู่ในรายการที่ถูกบันทึกไว้ก่อนหน้านี้ ให้ตอบจุดยอดนั้นแล้วจบการทำงาน

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
char p[100005];
char q[100005];
void add(char* x, int& len){
	for(int i = len-1; i >= 0; i--){
		if(x[i] == '0'){
			x[i] = '1';
			return;
		}else if(x[i] == '1'){
			x[i] = '2';
			return;
		}else{
			x[i] = '0';
		}
	}
	x[0] = '1';
	for(int i = 1; i <= len; i++){
		x[i] = '0';
	}
	x[len+1] = '\0';
	len++;
}
void fparent(char* x,int& len){
	add(x,len);
	x[len-1] = '\0';
	len--;
}
map<string,int> mp;
int main(){
```

```
	scanf("%s%s",p,q);
	int l1 = strlen(p);
	int l2 = strlen(q);
	int d1 = 0;
	int d2 = 0;
	mp[p] = d1;
	while(l1 != 0){
		fparent(p,l1);
		d1++;
		mp[p] = d1;
	}
	while(l2 != 0){
		if(mp.find(q) != mp.end()){
			printf("%d\n",mp[q]+d2-1);
			return 0;
		}
		fparent(q,l2);
		d2++;
	}
}
```

String

ในข้อนี้ เราจะต้องรองรับการทำงานของการแทรกและการย้อนสตริง ซึ่งสังเกตได้ว่ามีโครงสร้างข้อมูลที่สำคัญตัวหนึ่งที่รองรับการทำงานดังกล่าว นั่นคือ Binary Tree โดยเนื่องจากเราต้องการรับประกันการทำงานใน O(log |S|) เราจึงต้องใช้ Self Balancing Binary Tree ซึ่งในตัวอย่างจะใช้โครงสร้างข้อมูลที่เกิดจากการดัดแปลง Treap เมื่อต้องการ insert ณ ตำแหน่ง i ก็ต้อง split ส่วนก่อน i และส่วนหลัง i ออกมา แล้วค่อยๆ merge ตามลำดับ เมื่อต้องการ reverse ก็ต้องทำ Lazy Propagation บน Treap โดยการบันทึกไว้ว่าต้องการ reverse จุดยอดนี้ แล้วเมื่อเกิดการกระทำการใดๆที่จุดยอดใดก็ตาม ก็ต้อง update เพื่อให้การทำงานเสร็จสมบูรณ์

Implementation Code(C++)

```
#include <bits/stdc++.h>
using namespace std;
uniform_int_distribution<int> uid(INT_MIN,INT_MAX);
default_random_engine dre;
class trnode{
public:
	char key;
	int sz,prior;
	bool lazy;
	trnode *l,*r;
	trnode(char k = 0){
		key = k;
		sz = 1;
		lazy = false;
		prior = uid(dre);
		l = r = NULL;
	}
};
void upd(trnode* t){
	if(!t) return;
	t->sz = 1;
	if(t->l){
		t->sz += t->l->sz;
		if(t->lazy) t->l->lazy = !t->l->lazy;
	}
	if(t->r){
		t->sz += t->r->sz;
		if(t->lazy) t->r->lazy = !t->r->lazy;
	}
	if(t->lazy){
		swap(t->l,t->r);
		t->lazy = false;
	}
```

```
}
int size(trnode* t){
	return t ? t->sz : 0;
}
void split(trnode* t,int k,trnode*& l,trnode*& r){
	upd(t);
	if(!t) l = r = NULL;
	else if(size(t->l) >= k) split(t->l,k,l,t->l), r = t;
	else split(t->r,k-size(t->l)-1,t->r,r), l = t;
	upd(l);
	upd(r);
}
void merge(trnode*& t,trnode* l,trnode* r){
	upd(l);
	upd(r);
	if(!l || !r) t = l ? l : r;
	else if(l->prior > r->prior) merge(l->r,l->r,r), t = l;
	else merge(r->l,l,r->l), t = r;
	upd(t);
}
trnode* build(char* str){
	trnode* root = NULL;
	for(int i = 0; str[i]; i++){
		merge(root,root,new trnode(str[i]));
	}
	return root;
}
void printtreap(trnode* cur){
	upd(cur);
	if(!cur) return;
	printtreap(cur->l);
	printf("%c",cur->key);
	printtreap(cur->r);
}
char s[100005];
int main(){
	scanf("%s",s);
	int m;
	scanf("%d",&m);
	char cmd[2];
	int x,y;
	trnode* root = build(s);
	for(int i = 0; i < m; i++){
		scanf("%s",cmd);
		if(cmd[0] == 'R'){
			scanf("%d%d",&x,&y);
			trnode *ptr1,*ptr2,*ptr3;
			split(root,y,ptr1,ptr3);
			split(ptr1,x-1,ptr1,ptr2);
```

```
                ptr2->lazy = true;
                upd(ptr2);
                merge(root,ptr1,ptr2);
                merge(root,root,ptr3);
            }else{
                scanf("%d%s",&x,s);
                trnode *ptr1, *ptr2;
                ptr1 = ptr2 = NULL;
                split(root,x,ptr1,ptr2);
                merge(ptr2,build(s),ptr2);
                merge(root,ptr1,ptr2);
            }
        }
        printtreap(root);
        return 0;
}
```

Editorial by 1048576.

Problem by win11905, 1048576, and aquablitz11.

Grader system by Jittat Fakcharoenphol, Nattee Niparnan, win11905, and 1048576.

Special Thanks to the original grader system creators and contestants.