Portfólio de Introdução à Programação Competitiva

João Carlos Pandolfi Santana
joaopandolfi@gmail.com

Professores:
Jefferson O. Andrade
Flávio S. Lamas de Souza

14 de Julho de 2016

Conteúdo

1 Introdução

1.1 Começando: Super Fáceis

1.1.1 UVa 11172 - Relational Operator

Foi codificado sob a liguagem C.

```
#include <stdio.h>

int main(){
        int tam;
        long long a,b;
        scanf("%d",&tam);

        while(tam--){
                scanf("%lld %lld",&a,&b);
                if(a>b)
                        printf(">\n");
                else if(a < b)
                        printf("<\n");
                else
                        printf("=\n");
        }

        return 0;
}
```

1.2 Começando: Fáceis

1.2.1 UVa 11559 - Event Planning

Foi codificado sob a liguagem C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define readInt(a) scanf("%d",&a)
#define readString(s) scanf("%s",s)
#define left -1
#define right 1

int main(){
        int tests;
        char command[30];
        readInt(tests);
        while(tests--){
                int nCommands,i,result;
                readInt(nCommands);
                int commands[nCommands];
                i = result = 0;
                while(i<nCommands){
                        readString(command);

                        if(strcmp(command,"LEFT") == 0)
                                commands[i] = left;
                        else if(strcmp(command,"RIGHT") == 0)
```

```
                                commands[i] = right;
                        else{
                                int number;
                                readString(command);
                                readInt(number);
                                commands[i] = commands[number-1];
                        }
                        result += commands[i];
                        i++;
                }

                printf("%d\n",result);
        }


        return 0;
}
```

1.3 Começando: Médios

1.3.1 UVa 10420 - List of Conquests

Foi codificado sob a liguagem C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int main(){
        int tam,i,j,n,k,f,f2,v,a;
        char pal[2001][31];
        int nPal[2001];
        char aux[76];
        char cidade[31];
        i = n = a = 0;
        if(scanf("%d\n",&tam)!=1)
                return 0;
        memset(nPal,0,sizeof(nPal));
        while(i<tam){
                fgets(aux, sizeof(aux), stdin);
                j = 0;
                f = 0;
                v = 0;
                a = 0;
                memset(cidade,'\0',sizeof(cidade));
                while(j< 70 && aux[j] != '\0' && aux[j] != '\n'){
                        if(f != 1){
                                if(!isspace(aux[j])){
                                        cidade[a] = aux[j];
                                         a++;
                                         v = 1;
                                }
                                else if(v == 1){
                                        f = 1;
                                }
                        }else{
                                f2 = 1;
                                for(k =0 ; k<n; k++){
```

```
                                        if(strcmp(pal[k],cidade)==0){
                                                nPal[k]++;
                                                f2 = 0;
                                                break;
                                        }
                                }
                                if(f2 == 1){
                                        nPal[n]++;
                                        strcpy(pal[n],cidade);
                                        n++;
                                }
                                break;
                        }
                        j++;
                }
                if(f == 0){
                        nPal[n]++;
                        strcpy(pal[n],cidade);
                        n++;
                }

                i++;
        }

        for(i=0; i<n;i++){
                for(j = i;j<n;j++){
                        if(strcmp(pal[i],pal[j])>0){
                                strcpy(cidade,pal[j]);
                                strcpy(pal[j],pal[i]);
                                strcpy(pal[i],cidade);
                                f = nPal[i];
                                nPal[i] = nPal[j];
                                nPal[j] = f;
                        }
                }
        }

        for(i=0;i<n;i++){
                printf("%s %d\n",pal[i],nPal[i]);
        }
        return 0;
}
```

1.4 Problemas Ad Hoc

1.4.1 UVa 10921 - Find the Telephone

Foi codificado sob a liguagem C.

```
#include <stdio.h>

char correspondente(char val){
    if(val == '0' || val == '1' || val == '-')
        return val;
    if(val >= 'A' && val <= 'C')
        return '2';
    if(val >= 'D' && val <= 'F')
        return '3';
```

```
    if(val >= 'G' && val <= 'I')
        return '4';
    if(val >= 'J' && val <= 'L')
        return '5';
    if(val >= 'M' && val <= 'O')
        return '6';
    if(val >= 'P' && val <= 'S')
        return '7';
    if(val >= 'T' && val <= 'V')
        return '8';

    return '9';
}

int main(){
    char string[31];
    int i;
    while(scanf("%s",string) == 1){
        i=0;
        while(string[i] != '\0'){
            printf("%c",correspondente(string[i]));
            i++;
        }
        printf("\n");
    }
    return 0;
}
```

2 Bibliotecas e Estruturas de Dados

2.1 Estruturas de Dados Lineares

2.1.1 UVa 482 - Permutation Arrays

Foi codificado sob a liguagem C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define readInt(a) scanf("%d",&a)
#define readFloat(f) scanf("%f",&f)
#define readString(s) scanf("%s",s)
#define readLine(l) fgets(l, sizeof(l), stdin)
#define eraseVector(v,t) memset(v,'\0',t)

int main(){
        int cases;
        int index[500000];
        char result[5000][30];
        char linha[50000];
        char value[30];
        int tam;
        int count,i,k,j;
        if(scanf("%d\n",&cases)!= 1)
                return 0;
```

```
        while(cases--){

                count = tam = k = j = 0;
                eraseVector(value,30);
                readLine(linha);

                tam = strlen(linha);

                for(i =0; i<tam;i++){
                        if(linha[i] != ' ' && linha[i] != '\n' ){
                                value[k] = linha[i];
                                k++;
                        }
                        else{
                                index[count] = atoi(value)-1;
                                result[index[count]][0] = '\0';
                                count++;
                                k = 0;
                                eraseVector(value,30);
                        }
                }

                eraseVector(value,30);
                readLine(linha);
                k=0;

                tam = strlen(linha);

                for(i =0; i<tam;i++){
                        if(linha[i] != ' ' && linha[i] != '\n' ){
                                value[k] = linha[i];
                                k++;
                        }
                        else{
                                strcpy(result[index[j]],value);
                                j++;
                                k = 0;
                                eraseVector(value,30);
                        }
                }

                for(i = 0; i<count;i++){
                        if(result[i][0] != '\0')
                                printf("%s\n", result[i]);
                }

                printf("\n");
                scanf("\n");
        }

        return 0;
}
```

2.2 Estruturas de Dados Não Lineares

2.2.1 UVa 10295 - Hay Points

Foi codificado sob a liguagem C++.

```
#include <cstdio>
#include <iostream>
#include <string>
#include <map>
using namespace std;

#define gigante unsigned long long

int main() {
    int a, b;
    map<string, gigante> mapa;
    scanf("%d %d",&a,&b);

    int i=a;
    while(i--) {
        gigante v;
        string st;
        cin >> st >> v;
        mapa[st] = v;
    }

    i = b;
    while(i--) {
        gigante result = 0;
        string str;

        cin >> str;

        while ((str.length() != 1 || str[0] != '.')){
            result += mapa[str];
            cin >> str;
        }
        printf("%llu\n",result);

    }
    return 0;
}
```

2.3 Bibliotecas Próprias: Grafos

2.3.1 UVa 291 - The House Of Santa Claus

Foi codificado sob a liguagem C++.

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

int mapa [5][5];
int caminho[10];

void imprime(){
    int i;
```

```
    for(i = 0; i <= 8; i++ )
        printf("%d", caminho[i]);
    printf("\n");
}

void calculaRecursivo(int x, int y){
    int i;
    caminho[y] = x+1;

    if(y == 8){
        imprime();
        return;
    }

    for(i = 0; i < 5; i++)
        if(mapa[x][i]) {
            mapa[x][i] = mapa [i][x] = 0;
            calculaRecursivo(i, y + 1);
            mapa [x][i] = mapa [i][x] = 1;
        }
}

void inicializa(){
    mapa[0][0]= mapa[0][3] = 0;
    mapa[1][1] = mapa[1][3]= 0;
    mapa[2][2] = 0;
    mapa[3][0] = mapa[3][1] = mapa[3][3] = 0;
    mapa[4][4] = 0;
}

int main (){
    int i;
    for(i = 0; i < 5; i++ ) {
        for( int j = 0; j < 5; j++ ) {
            mapa[i][j] = 1;
        }
    }

    inicializa();
    calculaRecursivo(0, 0);
    return 0;
}
```

2.4 Bibliotecas Próprias: Conjuntos Disjuntos

2.4.1 UVa 10583 - Ubiquitous Religions

Foi codificado sob a liguagem C++.

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>

#include<vector>
#include<string>
#include<map>
```

```
using namespace std;

int lista[50001];

void preenche(int tam){
    for(int i=1; i<=tam; i++)
        lista[i] = i;
}

int verificaRecursivo(int pos){
    if(lista[pos] == pos)
        return pos;

    return lista[pos] = verificaRecursivo(lista[pos]);
}


int main(){
    int x,y,i,a,b,count;
    count = 0;

    scanf("%d",&x);
    scanf("%d",&y);
    while(!(x==0 && y==0)){
        map<int ,bool>hmap;

        preenche(x);

        for(i=0; i<y;i++ ){
            scanf("%d",&a);
            scanf("%d",&b);

            a = verificaRecursivo(a);
            b = verificaRecursivo(b);

            lista[a] = b;
        }

        for(int i=1; i<=x; i++)
            hmap[verificaRecursivo(i)] = true;

        printf("Case %d: %d\n", ++count, int(hmap.size()));

        scanf("%d",&x);
        scanf("%d",&y);
    }

    return 0;
}
```

2.5 Bibliotecas Próprias: Árvores de Segmentos

2.6 Bibliotecas Próprias: Árvores de Fenwick

3 Paradigmas de Resolução de Problemas

3.1 Pesquisa Completa

3.1.1 UVa 441 - Lotto

Foi codificado sob a liguagem C++.

```
#include<stdio.h>

int main(){
    int tam,flag;
    flag = 0;
    while(scanf("%d",&tam)){
        if(tam==0)
            break;

        int result[tam];

        if(flag == 1)
            printf("\n");
        else
            flag = 1;

        for(int i=0;i<tam;i++)
            scanf("%d",&result[i]);

        for(int x=0;tam-x>=6;x++)
            for(int y=x+1;tam-y>=5;y++)
                for(int z=y+1;tam-z>=4;z++)
                    for(int a=z+1;tam-a>=3;a++)
                        for(int b=a+1;tam-b>=2;b++)
                            for(int g=b+1;tam-g>=1;g++)
                                printf("%d %d %d %d %d %d\n",result[x],result[y],res

    }
    return 0;
}
```

3.2 Dividir e Conquistar

3.2.1 UVa 10611 - The Playboy Chimp

Foi codificado sob a liguagem C.

```
#include<stdio.h>

long long macacoThugLife[1000001];
int main(){
    long long i,j,k,maior,menor;
    long long quant,qtdData,lido;

    while(scanf("%lld",&quant)==1){
```

```
        for(i=0; i<quant; i++)
            scanf("%lld",&macacoThugLife[i]);

        scanf("%lld",&qtdData);

        i = qtdData;
        while(i--){
            maior=0;
            scanf("%lld",&lido);
            for(j=0; j<quant; j++){
                if(macacoThugLife[j]<lido)
                    maior=macacoThugLife[j];
                else
                    break;
            }

            if(maior==0)
                printf("X ");
            else
                printf("%lld ",maior);

            menor=0;
            k=quant;
            while(k--){
                if(macacoThugLife[k]>lido)
                    menor=macacoThugLife[k];
                else
                    break;
            }

            if(menor==0)
                printf("X\n");
            else
                printf("%lld\n",menor);
        }
    }
    return 0;
}
```

3.3 Algoritmos Gulosos

3.3.1 UVa 11292 - Dragon of Loowater

Foi codificado sob a liguagem C.

```
#include <stdio.h>


void sort(int val[], int tam) {
      int i, j, aux;
      for (i = 1; i < tam; i++) {
            j = i;
            while (j > 0 && val[j-1] > val[j]) {
                  aux = val[j];
                  val[j] = val[j-1];
                  val[j-1] = aux;
                  j--;
```

```
            }
        }
}

int main(){
        int diameters[20002];
        int talls[20002];
        int heads,knights;
        int i,j,cost,killCount;

        while(scanf("%d %d",&heads,&knights)){
                if(heads == 0 && knights == 0)
                        break;

                cost = killCount = 0;

                for(i=0; i<heads; i++)
                        scanf("%d",&diameters[i]);

                for(i=0; i<knights;i++)
                        scanf("%d",&talls[i]);

                sort(diameters,heads);
                sort(talls,knights);

                for(i=0;i<knights;i++){
                        for(j=killCount; j<heads;j++){
                                if(talls[i]>=diameters[j]){
                                        killCount++;
                                        cost+= talls[i];
                                        break;
                                }
                        }
                }

                if(killCount < heads){
                        puts("Loowater is doomed!");
                }
                else
                        printf("%d\n", cost);
        }


        return 0;
}
```

3.4 Programação Dinâmica

3.4.1 UVa 750 - 8 Queens Chess Problem

Foi codificado sob a liguagem C++.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int col[102],tam,linha,a,b;
```

```
void busca(int c){
        int x,y,count,flag;
        if ( c == 8 && col[b] == a){
                printf("%2d      %d",++linha, col[0] + 1 );
                for (x = 1; x < 8; x++)
                        printf(" %d", col[x] + 1);
                printf("\n");
        }

        for (y = 0; y < 8; y++){
                flag = 1;

                for (count = 0; count < c; count++)
                        if (col[count] == y || abs(count - c) == abs(col[count] - y)
                                flag = 0;
                                break;
                        }

                if(flag == 1){
                        col[c] = y;
                        busca(c + 1);
                }
        }
}

int main(){
        int i = 0;
        scanf("%d",&tam);
        while(i<tam){
                linha = 0;
                scanf("%d%d",&a,&b);
                a--;
                b--;
                printf("SOLN       COLUMN\n");
                printf(" #      1 2 3 4 5 6 7 8\n\n");
                busca(0);
                printf("\n");
                i++;
        }
        return 0;
}
```

4 Grafos

4.1 Pesquisa em Grafos

4.1.1 UVa 10116 - Robot Motion

Foi codificado sob a liguagem C++.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define visited(v) v>=0
#define MAP_TAM 10
#define readLine(l) fgets(l, sizeof(l), stdin)
```

```
int main(){
        int tam,qtdCommands,start,currC,currL;
        char map[11][11];
        int vis[11][11];
        int steps,i;
        while(1){
                scanf("%d %d %d\n",&tam,&qtdCommands,&start);

                if(tam == 0 && qtdCommands == 0 && start == 0)
                        break;

                currL = start-1;
                currC = steps = i = 0;
                while(i<tam){
                        readLine(map[i]);
                        scanf("\n");
                        for(int j=0 ;j<10;j++)
                                vis[i][j] = -1;
                        i++;
                }

                while(1){

                        if(currC >= tam || currC < 0 || currL >= qtdCommands || curr
                                printf("%d step(s) to exit\n", steps);
                                break;
                        }
                        else if(visited(vis[currC][currL])){
                                printf("%d step(s) before a loop of %d step(s)\n",vi
                                break;
                        }
                        vis[currC][currL] = steps;
                        switch(map[currC][currL]){
                                case 'N':
                                        currC--;
                                break;
                                case 'S':
                                        currC++;
                                break;
                                case 'W':
                                        currL--;
                                break;
                                case 'E':
                                        currL++;
                                break;
                        }
                        steps++;
                }
        }

        return 0;
}
```

4.2 Arvore Geradora Mínima

4.2.1 UVa 11733 - Airports

Foi codificado sob a liguagem C++.

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
using namespace std;

struct no{
    int f, g, h, j;
}N[100011];

int soma, n, m, contador;
int val[100011];
int pirulito[10011];

int bRecursiva( int x )   {
    if(x == pirulito[x])
        return x;

    return pirulito[x] = bRecursiva(pirulito[x]);
}

void Magia() {
    for ( int i = 0; i < m; ++i ) {
        int a = bRecursiva(N[i].f);
        int b = bRecursiva(N[i].g);
        if ( a != b ) {
            val[contador++] = N[i].h;
            pirulito[a] = b;
            soma += N[i].h;
        }
    }
}

bool checkb( int a, int b ) {
    return a > b;
}

bool checka( no a, no b ) {
    return a.h < b.h;
}

int main(){
    int A, tests,i, countPrint;
    scanf("%d", &tests);
    countPrint = 1;
    while(tests--) {
        int airport;
        scanf("%d %d %d", &n, &m, &A);

        for (i = 0; i<m; ++i )
            scanf("%d %d %d", &N[i].f, &N[i].g, &N[i].h);

        soma = airport = contador = 0;
```

```
        for (i = 0; i<=n; ++i )
            pirulito[i] = i;

        sort(N, N+m, checka);
        Magia();

        for (i = 1; i <= n; ++i )
            if ( i == bRecursiva(i) )
                airport++;

        soma += A*airport;

        sort( val, val+contador, checkb);

        for (i = 0; i < contador; ++i ) {
            if ( val[i] >= A ){
                soma = soma + A - val[i];
                airport++;
             }
            else
                break;
        }
        printf("Case #%d: %d %d\n", countPrint++, soma, airport);
    }
}
```