TRƯỜNG ĐẠI HỌC BÁCH KHOA

**KHOA: CÔNG NGHỆ THÔNG TIN**

BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

**ĐỀ THI CUỐI KỲ**

Tên học phần: TOÁN RỜI RẠC

Mã học phần: **……………………** Số tín chỉ: 3

Phương pháp đánh giá (\*): tự luận có giám sát Thời gian làm bài: 90 phút

Đề số: **Đ0002**

☐ Sinh viên được sử dụng tài liệu khi làm bài

**Họ tên:** ……………………… **Lớp**:………… …...……**MSSV**:………………...

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV\_HọTên.pdf và nộp bài thông qua MSTeam:

***Câu 1*** (*3 điểm*) Gọi Sn là số chuỗi độ dài *n* được hình thành từ các ký tự{A, B, C, D, E, F, 1, 2, 3}*.*

1. Đếm số chuỗi Sn  sao cho Sn chỉ chứa ký tự chữ

|  |
| --- |
| **# Trả lời:** Trình bày và dán kết quả vào bên dưới:  Gọi S=s1s2…sn-1sn là xâu chữ trên và an là số xâu S.  Có 6 khả năng xảy ra cho sn và số xâu S cho mỗi khả năng bằng số xâu s1s2…sn-1 chỉ chứa kí tự chữ. Trường hợp này, số sâu S bằng 6n. |

1. Lập hệ thức truy hồi để đếm số chuỗi Sn  sao cho Sn không chứa hai ký tự chữ kề nhau.

|  |
| --- |
| **# Trả lời:** Trình bày cách xây dựng hệ thức truy hồi vào đây:  #**Trả lời:** Xác định điều kiện ban đầu của hệ thức truy hồi vào đây:  Gọi S=s1s2…sn-1sn là xâu trên và an là số xâu S.  TH1. sn là ký tự số: có 3 khả năng xảy ra cho sn và số xâu S cho mỗi khả năng bằng số xâu s1s2…sn-1 không chứa hai ký tự chữ kề nhau và bằng an-1. Số sâu S có độ dài n sẽ bằng 3 lần số xâu s có độ dài n-1. Trường hợp này, số xâu S bằng 3a(n-1).  TH2. sn là ký tự chữ: để không có 2 ký tự chữ kề nhau nên sn-1 phải là ký tự số, do đó số xâu s có độ dài bằng n sẽ bằng 3x6 lần số xâu s có độ dài n-2. Trường hợp này, số sâu S bằng 18a(n-2).  => Hệ thức truy hồi: an = 3an-1 + 18an-2.  Đây là hệ thức truy hồi bậc 2 có 2 giá trị đầu là ao=1 và a1=9 |

1. Hãy giải hệ thức truy hồi trên câu b)

|  |
| --- |
| **# Trả lời:** Phương trình đặc trưng:  **# Trả lời:** Nghiệm phương trình đặc trưng ( nếu có):  Phương trình đặc trưng x^2= 3x + 18 có 2 nghiệm phân biệt là x = 6 và x = -3.  => an=b(6)^n+d(-3)^n  Dùng 2 giá trị đầu là a0=1 và a1=9. Ta tìm được b, d  => b= 4/3 và d=-1/3  Do đó, an= 4/3 \* 6^n + (-1/3)\*(-3)^n  **# Trả lời:** Nghiệm tổng quát (nếu có):  **# Trả lời:** Nghiệm của hệ thức truy hồi (nếu có): |

***Câu 2*** ( *3 điểm*) Cho tập X = {a, b, …, z }.

1. Viết chương trình liệt kê tập con k phần tử từ X (26) sử dụng phương pháp sinh.

|  |
| --- |
| **# Trả lời:** Dán code vào đây:  #include <stdio.h>  #include <dos.h>  #include <conio.h>  #define MAX 27  int S[MAX], c=0, n=26, k;  void print(){  int i;  printf("\n%3d: ",++c);  for (i=1; i<=k; i++) printf("%c ", S[i]-1+'a');  }  int main(){  int i,j;  printf("nhap k <= 26:"); scanf("%d",&k);  for (i=1; i<=k; i++) S[i]=i; print();  while (1){  i=k; while (i>0 && S[i]==n-k+i) i--;  if (i==0) break;  S[i]+=1;  for (j=i+1;j<=k;j++) S[j]=S[j-1]+1;  print();  }  }  **# Trả lời:** Dán kết quả minh họa vào đây với k = 4.    **# Trả lời:** Giải thích chi tiết cấu hình đầu tiên, cấu hình kết thúc, thuật toán sinh:  Cấu hình đầu tiên là a b c d, cấu hình cuối cùng w x y z.  Gải thích thuật toán:  Tìm S[i] bên phải nhất có thể tăng được (giá trị lớn nhất của vị trí i là (n-k+i))  Tăng S[i] với giá trị tăng nhỏ nhất (tìm phần tử lớn hơn phía sau nó)  Sau đó hoán đổi hai vị trí đó  Hạ cấu hình con S[i+1..k] xuống nhỏ nhất bằng cách đảo ngược đoạn từ i+1 đến k |

1. Viết chương trình liệt kê tất cả các tập con của X với số phân tử tối đa là 4 sử dụng kết quả của câu a)

|  |
| --- |
| **# Trả lời:** Dán code vào đây:  #include <stdio.h>  #include <dos.h>  #include <conio.h>  #define MAX 27  int S[MAX], c=0, n=26,k;  void print(){  int i;  printf("\n%3d: ",++c);  for (i=1; i<=k; i++) printf("%c ", S[i]-1+'a');  }  int main(){  for(k=1; k<=4; k++){  int i,j;  for (i=1; i<=k; i++) S[i]=i; print();  while (1){  i=k; while (i>0 && S[i]==n-k+i) i--;  if (i==0) break;  S[i]+=1;  for (j=i+1;j<=k;j++) S[j]=S[j-1]+1;  print();  }  }  }  **# Trả lời:** Dán kết quả thực thi minh họa vào đây: |

***Câu 3*** (*4 điểm*) Cho đồ thị liên thông, có trọng số như sau:

**E**

**B**

5

6

13

12

6

7

8

16

2

25

14

15

**A**

**G**

**D**

**C**

**F**

1. Dùng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh **A** đến đỉnh **G**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Trả lời:** Trình bày cách làm vào đây:   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | T | Vi | A | B | C | D | E | F | G | | {A..F} | - | 0 | oo | oo | oo | oo | oo | oo | | {B..F} | vA | 0 | 12 | 8 | oo | oo | oo | oo | | {B..F}\{C} | vC | - | 12 | 8 | 10 | oo | 34 | oo | | B,E,F | vD | - | 12 | - | 10 | 15 | 25 | oo | | E,F | vB | - | 12 | - | - | 15 | 25 | oo | | F | vE | - | - | - | - | 15 | 25 | 22 | | {rỗng} | vF | - | - | - | - | - | 25 | 22 |   **# Trả lời**: chỉ nhất đường đi ngắn nhất  => Đường đi ngắn nhất từ A->G là A->C->D->E->G  **# Trả lời:** độ đài đường đi ngắn nhất  Độ dài đường đi ngắn nhất là 22. |

1. Viết chương trình sử dụng thuật Dijkstra để tìm đường đi ngắn nhất từ đỉnh **A** đến đỉnh **G**.

|  |
| --- |
| **# Trả lời:** Dán code ở đây:  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  int Dijkstra(int u, int v, int a[][100], int p[], int n){  int D[100];  int T[100];// de danh dau phan tu  for (int i=1; i<=n; i++) D[i] = 1e9;// cho cac phan tu = vo cung  D[u] = 0;// chon vi tri xuat phat = 0  for (int i=1; i<=n ;i++) T[i] = 1; // danh dau cac phan tu chua duoc su dung  while (T[v]){ // chua di chuyen toi dich  int i=1;  while (T[i] == 0) i++;// khi T[i] da duoc su dung, tang i de tiep tuc duyet  for(int j = i+1; j<=n ;j++)  if(T[j] && D[j] < D[i])  i=j;// tim ra vi tri min de tiep tuc danh dau  T[i] = 0;// danh dau i da duoc dung  for (int j=1 ;j<=n ;j++)  if (T[j] && a[i][j]){// j ke i  if(D[i] + a[i][j] < D[j]){  D[j] = D[i] + a[i][j];  p[j] = i;  }  }  }  return D[v];// do dai duong di ngan nhat qua cac dinh  }  int main(){  int p[100], a[100][100];  int n; scanf("%d", &n);  for(int i=1; i<=n ;i++){  for(int j=1;j<=n;j++){  scanf("%d", &a[i][j]);  }  }  int ketqua = Dijkstra(1,7,a,p,n);  printf ("Ket qua la: %d", ketqua);  }  **# Trả lời:** Dán kết quả thực thi ở đây:    **# Trả lời:** Giải thích chi tiết thuật toán Dijkstra:  - Từ đỉnh gốc muốn chọn, khởi tạo khoảng cách tới chính nó là 0, khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là +oo. Ta được danh sách các khoảng cách tới các đỉnh  - Chọn đỉnh a có khoảng cách nhỏ nhất trong danh sách này và ghi nhận. Các lần sau sẽ không xét tới đỉnh này nữa  - Lần lượt xét các đỉnh kề b của đỉnh a. Nếu khoảng cách từ đỉnh gốc tới đỉnh b nhỏ hơn khoảng cách hiện tại đang được ghi nhận thì cập nhật giá trị và đỉnh kề a vào khoảng cách hiện tại của b  - Sau khi xét tất cả đỉnh kề b của đỉnh a. Lúc này ta được danh sách khoảng cách tới các điểm đã được cập nhật. Quay lại bước 2 với danh sách này. Thuâtj toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các điểm |

1. Dùng thuật toán Kruskal để tìm cây khung nhỏ nhất của đồ thị.

|  |
| --- |
| **# Trả lời:** Trình bày cách làm vào đây: |

1. Viết chương trình sử dụng thuật Kruskal để tìm cây khung nhỏ nhất của đồ thị.

|  |
| --- |
| **# Trả lời:** Dán code ở đây:  #include <stdio.h>  struct Edge {  int u, v;  int w;  };  typedef struct Edge Edge;  Edge edge[1001];  const int maxn = 1001;  int m = 9;  int n = 6;  int MST[maxn][maxn], Mat[maxn][maxn];  int parent[maxn], sz[maxn];  void make\_set() {  for (int i = 1; i <= n; i++) {  parent[i] = i;  sz[i] = 1;  }  }  int find(int v) {  if (v == parent[v])  return v;  return parent[v] = find(parent[v]);  }  void swap(int& a, int& b) {  int tmp = a;  a = b;  b = tmp;  }  bool Union(int a, int b) {  a = find(a);  b = find(b);  if (a == b)  return false;  if (sz[a] < sz[b])  swap(a, b);  parent[b] = a;  sz[a] += sz[b];  return true;  }  void Sort() {  for (int i = 0; i < m; i++)  for (int j = m - 1; j > i; j--)  if (edge[j - 1].w > edge[j].w) {  Edge tmp = edge[j - 1];  edge[j - 1] = edge[j];  edge[j] = tmp;  }  }  void kruskal() {  int d = 0;  make\_set();  Sort();  for (int i = 0; i < m; i++) {  if (Union(edge[i].u, edge[i].v)) {  MST[edge[i].u][edge[i].v] = MST[edge[i].v][edge[i].u] = edge[i].w;  d += edge[i].w;  }  }  printf("Do dai cay khung la: %d", d);  }  int main() {  printf("Nhap danh sach cac canh:\n");  for (int i = 0; i < m; i++) {  printf("Canh thu %d:\n", i + 1);  printf("Dinh dau: ");  scanf("%d", &edge[i].u);  printf("Dinh cuoi: ");  scanf("%d", &edge[i].v);  printf("Trong so: ");  scanf("%d", &edge[i].w);  printf("\n");  }  kruskal();  return 0;  }  **# Trả lời:** Dán kết quả thực thi ở đây:  **# Trả lời:** Giải thích chi tiết thuật toán Kruskal:  Ban đầu mỗi đỉnh là một cây riêng biệt, ta tìm cây khung nhỏ nhất bằng cách duyệt các cạnh theo trọng số từ nhỏ đến lớn, rồi hợp nhất các casyaas lại vs nhau  Cụ thể, giả sử cạnh đang xét nối 2 đỉnh u và v, nếu 2 đỉnh này nằm ở 2 cây khác nhau thì ta thêm cạnh này vào cây khung, đồng thời hợp nhất 2 cây chứa u và v  Ở đây, ta sử dụng cấu trúc [Disjoint Set](https://vietcodes.github.io/algo/dsu), độ phức tạp của toàn bộ thuật toán là O(MlogM) với M là số cạnh và đầu vào là danh sách cạnh có trọng số chứ không phải nhập ma trận trọng số như thuật toán Dijkstra. |

**Tổng cộng có: 3** câu

Đà Nẵng, ngày 08 tháng 05 năm 2023

|  |  |
| --- | --- |
| **GIẢNG VIÊN BIÊN SOẠN ĐỀ THI**  **(đã ký)** | **TRƯỞNG BỘ MÔN**  **(đã ký)** |