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 sátThời gian làm bài: 90 phút

Đề số: **Đ0001**

☐ Sinh viên được sử dụng tài liệu khi làm bài (tuy nhiên không được phép sử dụng ChatGPT)

—-------------------------------------------------------------------------------------------------------------------------------

**Họ tên:** Hà Đức Kiên **Lớp**: 23T\_DT1 **MSSV**:102230194

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*** (*2 điểm*) Cho 12 viên bi trong đó 4 viên bi xanh, 4 viên bi vàng và 4 viên bi đỏ, biết rằng các viên bi cùng màu là không phân biệt được.

1. Đếm số cách xếp 12 viên bi trên thành hàng ngang sao cho không có hai viên bi xanh xếp cạnh nhau.

|  |
| --- |
| **# Trả lời:** Trình bày cách giải và dán kết quả vào bên dưới:  Ta có xếp các viên bi theo cấu hình tổ hợp lặp về màu vàng và đỏ thì có số cách là:  N = 8!/(4!.4!) = 70 (cách)  Sao đó ta xếp vào sao cho giữa chúng sẽ tồn tại -1-1-1-1-1-1-1-1-1  Ta sẽ có số cách đặt sao cho các bi không trùng nhau là:  C39 = 84 (cách)  Vậy số cách xếp là:  N1= 84 x 70 = 5880 (cách) |

1. Đếm số cách xếp 12 viên bi trên thành hàng ngang sao cho không có bốn viên bi cùng màu xếp cạnh nhau.

|  |
| --- |
| **# Trả lời:** Trình bày cách giải và dán kết quả vào bên dưới:  4!.4!.4!= 13824 (cách) |

***Câu 2*** ( *2 điểm*) Cho Y là một dãy số và X là một hoán vị không lặp của Y

1. Hãy viết hàm nhập vào Y và đưa ra X (nếu có) với X là hoán vị nhỏ nhất của Y và lớn hơn Y.

|  |
| --- |
| **# Trả lời:** Dán code vào đây:  *#include* <stdio.h>  *#include* <stdlib.h>  *#define* *MAX* 20  *int* *S*[*MAX*], *c=*0, *n*;  *void* *print*();  *int* *Next*();  *int* *main*(){  *int* *i*;  *char* *input*[*MAX*];  *system*("COLOR F0");  *printf*("Nhap cau hinh ban dau: ");  *scanf*("%s", *input*);  *for* (*n* *=* 0; *input*[*n*] *!=* '\0'; *n++*);  *for* (*i* *=* 0; *i* *<* *n*; *i++*) {  *S*[*i* *+* 1] *=* *input*[*i*] *-* '0';  }  *printf*("Cau hinh hien tai: ");  *print*();  *if* (*Next*()) {  *printf*("Hoan vi ke tiep: ");  *print*();  } *else* {  *printf*("day la hoan vi cuoi cung.\n");  }  *return* 0;  }  *void* *print*() {  *int* *i*;  *for* (*i* *=* 1; *i* *<=* *n*; *i++*) {  *printf*("%d", *S*[*i*]);  }  *printf*("\n");  }  *int* *Next*() {  *int* *i* *=* *n* *-* 1, *j*, *tam*, *k*;  *while* (*i* *>* 0 *&&* *S*[*i*] *>* *S*[*i* *+* 1]) *i--*;  *if* (*i* *==* 0) *return* 0;  *j* *=* *n*;  *while* (*S*[*j*] *<* *S*[*i*]) *j--*;  *tam* *=* *S*[*i*];  *S*[*i*] *=* *S*[*j*];  *S*[*j*] *=* *tam*;  *j* *=* *i* *+* 1; *k* *=* *n*;  *while* (*j* *<* *k*) {  *tam* *=* *S*[*j*];  *S*[*j*] *=* *S*[*k*];  *S*[*k*] *=* *tam*;  *j++*;  *k--*;  }  *return* 1;  }  **# Trả lời:** Dán kết quả thực thi với Y = 4 3 1 2 |

1. Sử dụng phương pháp sinh viết chương trình liệt kê X, với X lớn hơn Y.

|  |
| --- |
| **# Trả lời:** Dán code vào đây:  *#include* <stdio.h>  *#include* <stdlib.h>  *#define* *MAX* 20  *int* *S*[MAX], c*=*0, n;  *void* *print*();  *int* *Next*();  *int* *main*(){  *int* i;  *char* *input*[MAX];  *system*("COLOR F0");  *printf*("Nhap cau hinh ban dau: ");  *scanf*("%s", input);  *for* (n *=* 0; *input*[n] *!=* '\0'; n*++*);  *for* (i *=* 0; i *<* n; i*++*) {  *S*[i *+* 1] *=* *input*[i] *-* '0';  }  *printf*("Cau hinh hien tai: ");  *print*();  *while* (*Next*()) {  *print*();  }  *printf*("Day la hoan vi cuoi cung.\n");  *return* 0;  }  *void* *print*() {  *int* i;  *for* (i *=* 1; i *<=* n; i*++*) {  *printf*("%d", *S*[i]);  }  *printf*("\n");  }  *int* *Next*() {  *int* i *=* n *-* 1, j, tam, k;  *while* (i *>* 0 *&&* *S*[i] *>* *S*[i *+* 1]) i*--*;  *if* (i *==* 0) *return* 0;  j *=* n;  *while* (*S*[j] *<* *S*[i]) j*--*;  tam *=* *S*[i];  *S*[i] *=* *S*[j];  *S*[j] *=* tam;  j *=* i *+* 1; k *=* n;  *while* (j *<* k) {  tam *=* *S*[j];  *S*[j] *=* *S*[k];  *S*[k] *=* tam;  j*++*;  k*--*;  }  *return* 1;  }  **# Trả lời:** Dán kết quả vào đây với trường hợp Y = 5 4 1 2 3 |

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

**E**

5

6

13

12

5

15

5

16

25

14

15

5

10

**B**

**G**

**D**

**A**

**H**

**F**

**C**

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Trả lời:** Trình bày cách giải bằng tay vào đây (yêu cầu trình bày dạng bảng):   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | T | vi | A | B | C | D | E | F | G | H | | A ...H | - | 0 | INF | INF | INF | INF | INF | INF | INF | | B ....H | A | 0\* | 12 | 5 | INF | INF | INF | INF | INF | | B, D...H | C | - | 10 | 5\* | 15 | INF | 30 | INF | INF | |  |  | - | 10\* | - | 15 | 16 | 30 | INF | INF | |  |  | - | - | - | 15\* | 16 | 30 | INF | INF | |  |  | - | - | - | - | 16\* | 30 | 31 | INF | |  |  | - | - | - | - | - | 30\* | 31 | 35 | |  |  | - | - | - | - | - | - | 31\* | 35 | |  |  | - | - | - | - | - | - | - | 35\* |   A->C->F->H = 35 |

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

|  |
| --- |
| **# Trả lời:** Dán code ở đây:  *#include* <stdio.h>  *#include* <string.h>  *#include* <stdbool.h>  *#define* *MAX* 20  *#define* *INF* 100000  *// Biến toàn cục*  *int* *n*;  *int* *a*[*MAX*][*MAX*];*// Ma trận kề*  *int* *D*[*MAX*][*MAX*];*// Bảng kết quả tính toán*  *int* *smin*[*MAX*];*// Lưu quãng đường ngắn nhất*  *void* *make\_D*();  *void* *display\_D*();  *void* *display\_D2*();  *void* *display\_Graph*();  *void* *min\_road*(*int* v);  *int* *main*() {  *// Đọc file*  *char* *filename*[20] *=* "test3";  *//printf("Nhap ten file (.inp): ");*  *//scanf("%s", filename);*  *strcat*(*filename*, ".inp");  FILE *\*file*;  *file* *=* *fopen*(*filename*, "r");  *if*(*file* *!=* *NULL*) {  *fscanf*(*file*, "%d", *&n*);  *// a[i][j] = 0 nếu không có cạnh (i, j)*  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) {  *for*(*int* *j* *=* 1; *j* *<=* *n*; *j++*) {  *fscanf*(*file*, "%d", *&a*[*i*][*j*]);  }  }  }  *// Đặt lại a[i][j] = INF nếu không có cạnh (i,j)*  *for* (*int* *i* *=* 1; *i* *<=* *n*; *i++*)  *for* (*int* *j* *=* 1; *j* *<=* *n*; *j++*)  *if* (*!a*[*i*][*j*]) *a*[*i*][*j*] *=* *INF*;  *printf*("Do thi:\n");  *display\_Graph*();  *printf*("\n");  *make\_D*();  *printf*("Bang ket qua:\n");  *display\_D*();  *int* *v*;*// Đỉnh cuối v*  *printf*("Nhap dinh muon di chuyen den: ");  *scanf*("%d", *&v*);  *min\_road*(*v*);  *fclose*(*file*);  *return* 0;  }  *void* *make\_D*() {  *bool* *T*[*MAX*];*// Nhãn cố định*  *// Ban đầu các đỉnh chưa được dán nhãn*  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) *T*[*i*] *=* 0;  *// Cài mặc định*  *D*[0][0] *=* 0;  *for*(*int* *i* *=* 2; *i* *<=* *n*; *i++*) {  *D*[0][*i*] *=* *INF*;  }  *int* *k* *=* 0;*// Số nhãn cố định*  *int* *current* *=* 1;*// Đỉnh đang xét hiện tại*  *smin*[*current*] *=* 0;*// Cập nhật đường đi ngắn nhất từ đỉnh đầu tới đỉnh current*  *// Bước lặp tiếp theo*  *T*[1] *=* 1;*// Đánh dấu Đỉnh 1*  *while*(*++k* *!=* *n*) {  *int* *min* *=* *INF*;*// Quãng đường ngắn nhất*  *int* *idx*;*// Đỉnh có quãng đường ngắn nhất*  *// Cập nhật đường đi tối ưu bước hiện tại*  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) {  *if*(*T*[*i*] *==* 0) {*// T[i] = 0: tức là đỉnh i chưa dán nhãn cố định*  *D*[*k*][*i*] *=* *D*[*k* *-* 1][*i*];  *if*(*D*[*k*][*i*] *>* *D*[*k* *-* 1][*current*] *+* *a*[*current*][*i*]) {  *D*[*k*][*i*] *=* *D*[*k* *-* 1][*current*] *+* *a*[*current*][*i*];  }  *// Tìm đỉnh quãng đường ngắn nhất bước hiện tại*  *if*(*min* *>* *D*[*k*][*i*]) {  *min* *=* *D*[*k*][*i*];  *idx* *=* *i*;  }  }  }  *current* *=* *idx*;*// Cập nhật đỉnh xét tiếp theo*  *T*[*current*] *=* 1;*// Dán nhãn cố định đỉnh current*  *smin*[*current*] *=* *min*;*// Cập nhật đường đi ngắn nhất từ đỉnh đầu tới đỉnh current*  }  }  *void* *display\_D*() {  *bool* *T*[*MAX*];  *// Bảng mặc định ban đầu*  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) *if*(*D*[0][*i*] *==* *INF*) *printf*("%c\t", 236);  *else* *printf*("%d\t", *D*[0][*i*]);  *printf*("\n");  *// Bảng được cập nhật*  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) {*//*  *for*(*int* *j* *=* 1; *j* *<=* *n*; *j++*) {  *if*(*D*[*i*][*j*] *==* 0) {  *if*(*T*[*j*] *==* 0) {  *printf*("%d\*\t", *D*[*i* *-* 1][*j*]);  *T*[*j*] *=* 1;  }  *else* *printf*("-\t");  }  *else* *if*(*D*[*i*][*j*] *==* *INF*)*printf*("%c\t", 236);  *else* *printf*("%d\t", *D*[*i*][*j*]);  }  *printf*("\n");  }  }  *void* *display\_D2*() {  *for*(*int* *i* *=* 1; *i* *<=* *n*; *i++*) {*//*  *for*(*int* *j* *=* 1; *j* *<=* *n*; *j++*) {  *printf*("%d\t", *D*[*i*][*j*]);  }  *printf*("\n");  }  }  *void* *display\_Graph*() {  *for* (*int* *i* *=* 1; *i* *<=* *n*; *i++*){  *for* (*int* *j* *=* 1; *j* *<=* *n*; *j++*) {  *if* (*a*[*i*][*j*] *<* *INF*) *printf*("%d\t", *a*[*i*][*j*]);  *else* *printf*("%c\t",236);  }  *printf*("\n");  }  }  *void* *min\_road*(*int* v) {  *int* *i* *=* v;  *int* *idx* *=* v;  *printf*("%d", v);  *while*(*i* *!=* 1) {  *if*(*D*[*i*][*idx*] *!=* 0 *&&* (*D*[*i*][*idx*] *!=* *D*[*i* *-* 1][*idx*])) {  *idx* *=* 1;  *while*(*true*) {  *if*(*D*[*i*][*idx*] *!=* 0 *||* (*D*[*i*][*idx*] *==* *D*[*i* *-* 1][*idx*])) *idx++*;  *else* *break*;  }  *printf*("<--%d", *idx*);  }  *i--*;  }  *printf*("<--1\n");  *printf*("Quang duong di ngan nhat: %d", *smin*[v]);  }  **# Trả lời:** Giải thích cách tìm kiếm đường đi của thuật toán Dijkstra:  **Bước 1: Chuẩn bị**   * **Đồ thị đầu vào**: Đồ thị có trọng số biểu diễn dưới dạng ma trận trọng số. * **Mảng khoảng cách (D)**: Lưu trữ khoảng cách ngắn nhất từ đỉnh xuất phát đến các đỉnh khác. * **Mảng tiền tố (p)**: Lưu trữ đỉnh liền trước trên đường đi ngắn nhất từ đỉnh xuất phát. * **Tập đỉnh chưa được xét (T)**: Lưu trữ các đỉnh chưa được xét.   **Bước 2: Khởi tạo**   * **Khởi tạo khoảng cách ban đầu**: Đặt khoảng cách từ đỉnh xuất phát u đến chính nó là 0, và các khoảng cách khác là vô cùng (INF). * **Khởi tạo tập đỉnh chưa được xét**: Ban đầu, tất cả các đỉnh đều chưa được xét.   **Bước 3: Tìm đỉnh có khoảng cách ngắn nhất**   * **Duyệt qua các đỉnh chưa được xét**: Tìm đỉnh có khoảng cách ngắn nhất trong tập các đỉnh chưa được xét. * **Đánh dấu đỉnh này đã được xét**: Cập nhật đỉnh này là đã được xét.   **Bước 4: Cập nhật khoảng cách**   * **Duyệt qua các đỉnh kề của đỉnh vừa chọn**: Kiểm tra các đỉnh kề của đỉnh vừa được xét và cập nhật khoảng cách nếu tìm được đường đi ngắn hơn. * **Cập nhật mảng tiền tố**: Lưu lại đỉnh trước đó trên đường đi ngắn nhất.   **Bước 5: Lặp lại**   * **Lặp lại quá trình trên**: Tiếp tục lặp lại cho đến khi đỉnh đích v được xét.   **Bước 6: Kết quả**   * **Kết quả**: Khoảng cách ngắn nhất từ đỉnh xuất phát u đến đỉnh đích v và đường đi ngắn nhất. |

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

|  |
| --- |
| **# Trả lời:** Trình bày cách làm bằng tay vào đây:  A đến C nhỏ nhất là 5  C tới B nhỏ nhất là 5  B tới E nhỏ nhất là 6  E tới D nhỏ nhất là 5  E tới F nhỏ nhất là 14  F đến H nhỏ nhất là 5  H đến G nhỏ nhất là 10  Trọng số T = 50.    B  A  G  E  D  H  F  C |

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>  *#include* <stdlib.h>  *#include* <limits.h>  *#include* <string.h>  *#define* *V* 30  *int* *parent*[*V*];  *int* *rank*[*V*];  *// Find set of vertex i (with path compression)*  *int* *find*(*int* i) {  *if* (*parent*[i] *!=* i) {  *parent*[i] *=* *find*(*parent*[i]);  }  *return* *parent*[i];  }  *// Does union of i and j using union by rank*  *void* *union1*(*int* i, *int* j) {  *int* *root\_i* *=* *find*(i);  *int* *root\_j* *=* *find*(j);  *if* (*root\_i* *!=* *root\_j*) {  *if* (*rank*[*root\_i*] *<* *rank*[*root\_j*]) {  *parent*[*root\_i*] *=* *root\_j*;  } *else* *if* (*rank*[*root\_i*] *>* *rank*[*root\_j*]) {  *parent*[*root\_j*] *=* *root\_i*;  } *else* {  *parent*[*root\_j*] *=* *root\_i*;  *rank*[*root\_i*]*++*;  }  }  }  *// Finds MST using Kruskal's algorithm*  *void* *kruskalMST*(*int* cost[][*V*], *int* n) {  *int* *mincost* *=* 0; *// Cost of min MST.*  *// Initialize sets of disjoint sets.*  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *parent*[*i*] *=* *i*;  *rank*[*i*] *=* 0;  }  *// Include minimum weight edges one by one*  *int* *edge\_count* *=* 0;  *while* (*edge\_count* *<* n *-* 1) {  *int* *min* *=* *INT\_MAX*, *a* *=* *-*1, *b* *=* *-*1;  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*find*(*i*) *!=* *find*(*j*) *&&* cost[*i*][*j*] *<* *min*) {  *min* *=* cost[*i*][*j*];  *a* *=* *i*;  *b* *=* *j*;  }  }  }  *union1*(*a*, *b*);  *printf*("Edge %d: (%d, %d) cost: %d\n", *++edge\_count*, *a*, *b*, *min*);  *mincost* *+=* *min*;  }  *printf*("\nMinimum cost spanning tree = %d\n", *mincost*);  }  *// driver program to test above function*  *int* *main*() {  *int* *i*, *j*, *n*;  *int* *Graph*[*V*][*V*];  *char* *fn*[101];  *printf*("Enter the file name (\*.INP): ");  *if* (*fgets*(*fn*, *sizeof*(*fn*), *stdin*) *!=* *NULL*) {  *size\_t* *len* *=* *strlen*(*fn*);  *if* (*len* *>* 0 *&&* *fn*[*len* *-* 1] *==* '\n') {  *fn*[*len* *-* 1] *=* '\0';  }  }  *if* (*!strchr*(*fn*, '.')) {  *strcat*(*fn*, ".INP");  }  FILE *\*file* *=* *fopen*(*fn*, "r");  *if* (*file* *==* *NULL*) {  *perror*("Error opening file");  *return* *EXIT\_FAILURE*;  }  *fscanf*(*file*, "%d", *&n*);  *for* (*i* *=* 0; *i* *<* *n*; *i++*) {  *for* (*j* *=* 0; *j* *<* *n*; *j++*) {  *fscanf*(*file*, "%d", *&Graph*[*i*][*j*]);  *if* (*Graph*[*i*][*j*] *==* 0) {  *Graph*[*i*][*j*] *=* *INT\_MAX*;  }  }  }  *fclose*(*file*);  *printf*("n = %d\n", *n*);  *for* (*i* *=* 0; *i* *<* *n*; *i++*) {  *for* (*j* *=* 0; *j* *<* *n*; *j++*) {  *if* (*Graph*[*i*][*j*] *!=* *INT\_MAX*) {  *printf*("%d\t", *Graph*[*i*][*j*]);  } *else* {  *printf*("INF\t");  }  }  *printf*("\n");  }  *// Print the solution*  *kruskalMST*(*Graph*, *n*);  *return* 0;  }  **# Trả lời:** Giải thích cách xây dựng cây khung bé nhất của thuật toán Kruskal:  **Bước 1: Chuẩn bị**   * **Đồ thị đầu vào**: Đồ thị có trọng số biểu diễn dưới dạng danh sách cạnh hoặc ma trận trọng số. * **Danh sách cạnh**: Chuyển đổi đồ thị đầu vào thành danh sách các cạnh kèm theo trọng số.   **Bước 2: Sắp xếp các cạnh**   * **Sắp xếp**: Sắp xếp các cạnh theo thứ tự trọng số tăng dần. Điều này giúp đảm bảo rằng mỗi bước tiếp theo luôn chọn được cạnh nhỏ nhất.   **Bước 3: Khởi tạo các cấu trúc dữ liệu**   * **Cấu trúc Union-Find**: Để kiểm tra và hợp nhất các tập hợp đỉnh một cách hiệu quả, ta sử dụng cấu trúc dữ liệu Union-Find. Cấu trúc này bao gồm hai mảng:   + parent[]: Để lưu cha của mỗi đỉnh, giúp tìm gốc của một đỉnh.   + rank[]: Để lưu độ cao của cây, giúp tối ưu quá trình hợp nhất các tập hợp.   **Bước 4: Duyệt qua các cạnh và xây dựng MST**   * **Duyệt các cạnh**: Lần lượt xét từng cạnh theo thứ tự đã sắp xếp. * **Kiểm tra chu trình**: Với mỗi cạnh (u, v), kiểm tra xem u và v có thuộc cùng một tập hợp không bằng cách sử dụng hàm find(). Nếu không, thêm cạnh này vào MST và hợp nhất hai tập hợp chứa u và v bằng hàm union\_set(). * **Thêm cạnh vào MST**: Nếu cạnh không tạo chu trình, thêm cạnh đó vào cây khung.   **Bước 5: Kết thúc**   * **Kết quả**: Sau khi duyệt hết các cạnh hoặc khi đã có đủ n−1cạnh (với n là số đỉnh của đồ thị), cây khung nhỏ nhất đã được tìm thấy. |

***Câu 4*** ( *2 điểm*) Cho biểu thức Boole:

1. Xây dựng bảng chân trị của biểu thức Boole trên

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Trả lời:** Dán bảng chân trị vào đây:   |  |  |  |  | | --- | --- | --- | --- | | X | Y | Z | E (X,Y,Z) | | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 | | 1 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | |

1. Sử dụng 11 tính chất của đại số Boole tối thiểu của .

|  |
| --- |
| **# Trả lời:** Dán các bước biến đổi vào đây: |
| E(x,y,z) = (xyz + xyz )+ (xyz + xyz) + (xyz + xyz) = xy + xy + xz = y + xz |

**Tổng cộng có: 4** câu

Đà Nẵng, ngày 25 tháng 05 năm 2024

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