I, trắc nghiệm

1. D
2. A
3. A
4. D
5. C
6. C
7. C
8. B
9. B
10. B

II, tự luận

Câu 1:

#include <bits/stdc++.h>

using namespace std;

void nhap(int a[],int n){

for (int i=0;i<n;i++) cin>>a[i];

}

void timkiem(int a[],int n,int k){

for(int i=0;i<n;i++){

for (int j=i;j<n;j++){

if (a[i]+a[j]==k && i!=j) cout<<a[i]<<" "<<a[j]<<endl;

}

}

}

int main(){

int n,a[100000],k;

cin>>n;

nhap(a,n);

cin>>k;

timkiem(a,n,k);

return 0;

}

Câu 2:

#include <bits/stdc++.h>

using namespace std;

int main(){

int m,n,k,q;

int x,y,z;

int a,b;

int mang[100][100];

cin>>m>>n>>k>>q;

while (k--){

cin>>x>>y>>z;

mang[x][y] = z;

}

while (q--){

cin>>a>>b;

int sum = 0;

sum=mang[a][b+1]+mang[a+1][b]+mang[a][b-1]+mang[a-1][b]+mang[a+1][b+1]+mang[a-1][b-1]+mang[a+1][b-1]+mang[a-1][b+1];

cout<<sum<<endl;

}

return 0;

}

Câu 3:

void sapxep(int a[], int n) {

    for (int i = 0; i < n - 1; i++) {

        for (int j = i + 1; j < n; j++) {

            if (a[i] > a[j]) {

                swap(a[i], a[j]);

            }

        }

    }

}

|  |  |  |
| --- | --- | --- |
| **Thuật toán** | **Ưu điểm** | **Nhược điểm** |
| **Bubble sort** | - Dễ hiểu, dễ cài đặt.  - Có thể dừng sớm nếu mảng đã sắp xếp | - Chậm nhất trong 3 thuật toán (O(n²)).  - Không hiệu quả với dữ liệu lớn. |
| **Selection sort** | - Không bị ảnh hưởng bởi trạng thái ban đầu của mảng (luôn chạy O(n²)).  - Ít hoán đổi phần tử hơn Bubble Sort. | - Cũng có độ phức tạp O(n²).  - Hiệu suất kém hơn Insertion Sort nếu dữ liệu gần như đã sắp xếp. |
| **Insertion Sort** | - Nhanh hơn Bubble Sort và Selection Sort khi mảng gần như đã sắp xếp (có thể đạt O(n)).  - Hiệu quả với dữ liệu nhỏ. | - Không phù hợp cho dữ liệu lớn (vẫn O(n²) trong trường hợp xấu). |

Câu 4

Bước 1: sắp xếp mảng theo thứ tự tăng dần

Bước 2: xác định vị trí Mid ( vị trí giữa của mảng) mid=n/2

Bước 3: So sánh Mid với số cần tìm(K)

* Nếu k=a[mid] trả về vị trí mid
* Nếu K<a[mid] tìm kiếm ở nửa trái (mid-1)
* Nếu k>a[mid] tìm kiếm ở nửa phải (mid+1)