Example codes

Sunday, November 21, 2021 12:48 AM

```cpp
using namespace std;

void BFS(int arr[][7], int node, int n)
{
    int visited[n] = {0};
    queue<int> q;
    int v = node;

    printf("%d ", v);
    visited[v] = 1;
    q.push(v);

    while(!q.empty())
    {
        v = q.front();
        q.pop();
        for(int i = 1; i < n; i++){
            if(arr[v][i] == 1 && visited[i] == 0){
                printf("%d ", i);
                visited[i] = 1;
                q.push(i);
            }
        }
    }
    printf("\n");
}

int main()
```

```cpp
// Name : Md Noor E Musa
// ID : 18CSE033

// C++ program for implementation of Depth-first search

#include <iostream>
using namespace std;

void DFS(int arr[][7], int node, int n)
{
    static int visited[7] = {0};
    if(visited[node] == 0){
        printf("%d ", node);
        visited[node] = 1;

        for(int i = 1; i < n; i++)
            if(arr[node][i] && visited[i] == 0)
                DFS(arr, i, n);
    }
}

int main()
{
    int g[7][7] = {
        {0, 0, 0, 0, 0, 0, 0},
        {0, 0, 1, 1, 0, 0, 0},
        {0, 1, 0, 0, 1, 0, 0},
        {0, 1, 0, 0, 1, 0, 0},
        {0, 0, 1, 1, 0, 1, 1},
```

```cpp
    return mx;
}

void countingSort(int arr[], int n, int p) {
    int out[n], count[10];

    for(int i = 0; i < 10; i++) count[i] = 0;

    for(int i = 0; i < n; i++) count[(arr[i]/p)%10]++;

    for(int i = 1; i < 10; i++) count[i] += count[i-1];

    for(int i = n-1; i >= 0; i--){
        out[count[(arr[i]/p)%10]-1] = arr[i];
        count[(arr[i]/p)%10]--;
    }

    for(int i = 0; i < n; i++) arr[i] = out[i];
}

void radixsort(int arr[], int n) {
    int mx = getMax(arr, n);
    for(int p = 1; mx/p>0; p*=10){
        countingSort(arr, n, p);
    }
}

void printArr(int arr[], int n)
{
```

```cpp
void merge(int arr[], int l, int mid, int h)
{
    int i = l, j = mid+1, k = l, ara[h+1];

    while(i <= mid && j <= h) {
        if(arr[i] < arr[j])
            ara[k++] = arr[i++];
        else
            ara[k++] = arr[j++];
    }

    while(i <= mid) ara[k++] = arr[i++];
    while(j <= h) ara[k++] = arr[j++];

    while(l <= h){
        arr[l] = ara[l];
        l++;
    }
}

void mergeSort(int arr[], int l, int h)
{
    if(l < h) {
        int m = l+(h-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, h);
        merge(arr, l, m, h);
    }
```

```cpp
// C++ program for implementation of Selection Sort

#include <iostream>
using namespace std;

void selectionSort(int arr[], int n)
{
    for(int i = 0; i < n-1; i++){
        int mi = i;

        for(int j = i+1; j < n; j++){
            if(arr[mi] > arr[j])
                mi = j;
        }

        if(mi != i)
            swap(arr[mi], arr[i]);
    }
}

void printArr(int arr[], int n)
{
```

```cpp
// C++ program for implementation of Insertion Sort

#include <iostream>
using namespace std;

void insertionSort(int arr[], int n)
{
    for(int i = 1; i < n; i++){
        int j = i-1;
        int temp = arr[i];

        while(j > -1 && temp < arr[j]){
            arr[j+1] = arr[j];
            j--;
        }

        arr[j+1] = temp;
    }
}

void printArr(int arr[], int n)
{
```

```cpp
void heapify(int arr[], int n, int i)
{
    int b = i, l = i*2+1, r = i*2+2;

    if(l < n && arr[l] > arr[b])
        b = l;

    if(r < n && arr[r] > arr[b])
        b = r;

    if(b != i){
        swap(arr[b], arr[i]);
        heapify(arr, n, b);
    }
}

void heapSort(int arr[], int n)
{
    for(int i = n/2-1; i >= 0; i--)
        heapify(arr, n, i);

    for(int i = n-1; i > 0; i--){
        swap(arr[0], arr[i]);
        heapify(arr, i, 0);
    }
}
```

```cpp
// C++ program for implementation of Bubble Sort

#include <iostream>
using namespace std;

void bubbleSort(int arr[], int n)
{
    for(int i = 0; i < n-1; i++){
        bool swpped = false;
        for(int j = 0; j < n-1-i; j++){
            if(arr[j+1] < arr[j]){
                swap(arr[j+1], arr[j]);
                swpped = true;
            }
        }
        if(swpped == false) break;
    }
}

void printArr(int arr[], int n)
{
```

```cpp
    int deleteNode(int index)
    {
        if(index < 0){
            cout << "index must be grater than or equal to 0." << endl;
            exit(1);
        }
        int ele = -1;
        Node *p = first, *q = nullptr;
        if(index == 0){
            first = first->next;
            ele = p->data;
            delete p;
            p = nullptr;
        }
        else
        {
            for(int i = 0; i < index && p; i++){
                q=p;
                p=p->next;
            }

            if(p){
                q->next = p->next;
                ele = p->data;
                delete p;
                p = nullptr;
            }
            else
            {
                cout << "Linked List is not long enough to delete node at index: " << index << endl;
                exit(1);
            }
        }

        return ele;
```

```cpp
    }

    bool insertNode(int data, int index)
    {
        if(index < 0){
            cout << "index must be grater than or equal to 0." << endl;
            return false;
        }

        Node* t = new Node(data);

        if(index == 0) {
            t->next = first;
            first = t;
            return true;
        }

        Node* prev = first;
        for(int i = 1; i < index && prev; i++)
            prev = prev->next;

        if(prev){
            t->next = prev->next;
            prev->next = t;
            return true;
        }

        cout << "Linked List is not long enough to insert node at index: " << index << endl;
        return false;
    }

    int deleteNode(int index)
    {
        if (index < 0) {
            cout << "index must be grater than or equal to 0." << endl;
```

```cpp
        first = nullptr;
    }

    bool insertNode(int data, int index)
    {
        if (index < 0) {
            cout << "index must be grater than or equal to 0." << endl;
            return false;
        }

        Node *temp = new Node(data);

        if(index == 0)
        {
            temp->next = first;
            first = temp;
            return true;
        }

        Node *prev = first;
        for (int i = 1; i < index && prev; i++)
            prev = prev->next;

        if(prev)
        {
            temp->next = prev->next;
            prev->next = temp;
            return true;
        }

        cout << "Linked List is not long enough to insert node at index: " << index << endl;
        return false;
    }

    int deleteNode(int index)
```

```cpp
struct Node* deleteNode(struct Node *p, int item)
{
    if(p == NULL)
        return NULL;

    if(!p->l && !p->r)
    {
        if(p == root)
            root = NULL;
        free(p);
        return NULL;
    }

    if(item < p->data)
        p->l = deleteNode(p->l, item);
    else if(item > p->data)
        p->r = deleteNode(p->r, item);
    else
    {
        struct Node *q;
        if(height(p->l) > height(p->r))
        {
            q = inpre(p->l);
            p->data = q->data;
            p->l = deleteNode(p->l, q->data);
        }
        else
        {
            q = insuc(p->r);
            p->data = q->data;
            p->r = deleteNode(p->r, q->data);
        }
    }

    return p;
```

```cpp
struct Node* insertNode(struct Node *t, int data)
{
    if(t == NULL)
    {
        t = (struct Node*)malloc(sizeof(struct Node));
        t->data = data;
        t->l = t->r = NULL;
        return t;
    }
    if(t->data < data)
        t->r  = insertNode(t->r, data);
    else if(t->data > data)
        t->l = insertNode(t->l, data);
    return t;
}

void search(struct Node *t, int item)
{
    if(t)
    {
        if(t->data < item)
            search(t->r, item);
        else if(t->data > item)
            search(t->l, item);
        else
        {
            printf("Found %d\n", item);
            return;
        }
    }
    else printf("NOT Found %d\n", item);
}
```

```cpp
int dist[V]; // Output/retuslt array
void Dijkstra(int graph[V][V], int start)
{
    priority_queue<pi, vector<pi>, greater<pi> > pq;
    bool visited[V];

    for(int i = 0; i < V; i++){
        dist[i] = inf;
        visited[i] = false;
    }

    dist[start] = 0;
    pq.push(make_pair(0, start));

    while(!pq.empty()) {
        pi node = pq.top();
        pq.pop();

        visited[node.name] = true;
        if(dist[node.name] < node.dist) continue;

        for(int i = 0; i < V; i++) {
            if(graph[node.name][i] > 0 && visited[i] == false) {
                int new_dist = graph[node.name][i] + node.dist;
                if(new_dist < dist[i]) {
                    dist[i] = new_dist;
                    pq.push(make_pair(new_dist, i));
                }
            }
        }
    }
}
```

```cpp
int partition(int arr[], int l, int h)
{
    int pivot = arr[l], i = l, j = h;

    do {
        do { i++; } while(pivot >= arr[i]);
        do { j--; } while(pivot < arr[j]);
        if(i < j) swap(arr[i], arr[j]);
    }while(i < j);
    swap(arr[l], arr[j]);
    return j;
}

void quickSort(int arr[], int l, int h)
{
    if(l < h) {
        int p = partition(arr, l, h);
        quickSort(arr, l, p);
        quickSort(arr, p+1, h);
    }
}
```