[자료구조 응용 기말 대비]

전자공학부 이은찬

[기말테스트 범위]

DS 14~DS 22 (총 9개)

Ch5.트리--------------------------

DS14. Implements Tree and Its Traversal

DS15. Max Heap and Its Traversal

DS16. Binary Search Tree and Its Traversal

DS17.Winner Tree and Its Traversal

Ch6.그래프-----------------------

DS18.DFS and BFS with Its Adj Matrix and Adj List

DS19.Kruskal Algorithm with MST(Minimum Spanning Tree)

DS20.AOV networks(Active on Vertex)

Ch7.정렬---------------------------

DS21.QuickSort and HeapSort

Ch8.해싱---------------------------

DS22.Hashing Simple Example

[개요]

<코드 난이도 분류(주관적)>

EASY MIDIUM HARD

EASY: DS14, DS17, DS22

MIDIUM: DS15, DS16

HARD: DS18, DS19, DS20, DS21

[DS14]

Implements Tree and Its Traversal

트리 구성 코드

```
typedef struct node *treePointer;

typedef struct node {
char data;
treePointer leftChild, rightChild;
};

void inorder(treePointer ptr) {
```

```
if (ptr) {
inorder(ptr->leftChild);
printf("%c", ptr->data);
inorder(ptr->rightChild);
}
}

void postorder(treePointer ptr) {
if (ptr) {
postorder(ptr->leftChild);
postorder(ptr->rightChild);
printf("%c", ptr->data);

}
}

void preorder(treePointer ptr) {
if (ptr) {
printf("%c", ptr->data);
preorder(ptr->leftChild);
preorder(ptr->rightChild);

}
}
```

[DS15] HEAP 삽입,삭제 코드 등

->나름 복잡(코드 FULL VER)

```
#define HEAP_FULL(n) (n == MAX_ELEMENTS-1)
#define HEAP_EMPTY(n) (!n)
typedef struct {
    int key;
}element;
typedef struct node *treePointer;

typedef struct node{
    int data;
    treePointer left, right;
};
treePointer tree[MAX_ELEMENTS];
element heap[MAX_ELEMENTS];
int n = 0;

void push(element item, int n) {
    int i;
    if (HEAP_FULL(n)) {
        fprintf(stderr, "The heap is full.\n");
        exit(EXIT_FAILURE);
    }
    i = ++(n);
    while ((i != 1) && (item.key > heap[i / 2].key)) {
        heap[i] = heap[i / 2];
        i /= 2;
    }
    heap[i] = item;
}
```

```
element pop(int n) {
    int parent, child;
    element item, temp;
    if (HEAP_EMPTY(n)) {
        fprintf(stderr, "The heap is full.\n");
        exit(1);
    }
    item = heap[1];
    temp = heap[(n)--];
    parent = 1;
    child = 2;
    while (child <= n) {
        if ((child <n) && (heap[child].key < heap[child + 1].key))
            child++;
        if (temp.key >= heap[child].key) break;

        heap[parent] = heap[child];
        parent = child;
        child *= 2;
    }
    heap[parent] = temp;
    return item;
}

void inorder(treePointer ptr) {
    if (ptr) {
        inorder(ptr->left);
        printf("%d ", ptr->data);
        inorder(ptr->right);
    }
}

int main() {
    int x=0, siz;
    FILE *f;
    fopen_s(&f, "input.txt", "r");
    int i = 1;
    fscanf_s(f, "%d", &x);
    element temp;
    temp.key = x;
    push(temp, 0);

    for (i = 0; i < MAX_ELEMENTS; i++) {
        tree[i] = malloc(sizeof(tree[i]));

    }

    i = 1;
    while (!feof(f)) {
        element root;
        fscanf_s(f, "%d", &x);
        root.key = x;
        tree[i+1]->data = x;
        push(root, i++);
    }
    siz = i;

    for(i=1;i<=siz;i++)
```

```
tree[i]->data = heap[i].key;


//left,right 처리
for (i = 1; i <= siz; i++) {
    if (i * 2 <= siz)tree[i]->left = tree[i * 2];
    else tree[i]->left = NULL;


    if ((i * 2) + 1 <= siz)tree[i]->right = tree[i * 2 + 1];
    else tree[i]->right = NULL;
}




//출력********************************************
//#1
printf("Level Order: ");
for (i = 1; i <= siz; i++)
    printf("%d ", heap[i].key);
printf("\nInorder: ");
inorder(tree[1]);
printf("\n");
//


//#2
pop(siz);


siz--;
for (i = 1; i <= siz; i++)
    tree[i]->data = heap[i].key;
```

```
//left,right 처리
for (i = 1; i <= siz; i++) {
    if (i * 2 <= siz)tree[i]->left = tree[i * 2];
    else tree[i]->left = NULL;

    if ((i * 2) + 1 <= siz)tree[i]->right = tree[i * 2 + 1];
    else tree[i]->right = NULL;
}

printf("Level Order: ");
for (i = 1; i <= siz; i++)
    printf("%d ", heap[i].key);

printf("\nInorder: ");
inorder(tree[1]);
printf("\n");
//
//#3
pop(siz);

siz--;
for (i = 1; i <= siz; i++)
    tree[i]->data = heap[i].key;

//left,right 처리
for (i = 1; i <= siz; i++) {
    if (i * 2 <= siz)tree[i]->left = tree[i * 2];
```

```
        else tree[i]->left = NULL;


        if ((i * 2) + 1 <= siz)tree[i]->right = tree[i * 2 + 1];
        else tree[i]->right = NULL;
    }


    printf("Level Order: ");
    for (i = 1; i <= siz; i++)
        printf("%d ", heap[i].key);


    printf("\nInorder: ");
    inorder(tree[1]);
    printf("\n");
    //


    return 0;
}
```

[DS16] Binary Search Tree

탐색 구현

```
void process(treePointer tp, int k) {
    treePointer temp1, temp2;


    while (tp) {
        temp1 = malloc(sizeof(temp1));
        temp1->data = k;
        temp1->left = temp1->right = NULL;
        if (k < tp->data){

            if (!tp->left) {
                tp->left = temp1; break;
            }
            else tp = tp->left;
        }
        else if (k > tp->data) {
            if (!tp->right) {
                tp->right = temp1;
                break;
            }
            else {
                tp = tp->right;
            }
        }
        else break;
    }
}

int search(treePointer tr, int key) {
    while (tr) {
        if (key == tr->data) return key;
        else if (key < tr->data) tr = tr->left;
        else  tr = tr->right;
    }
     return -99999;
}
```

[DS17]

2D array alloc 구현

```
//2D ARRAY ALLOC  arr[n][MAX] =all set 0
int **arr;
arr= (int **)calloc(n, sizeof(int *));
   for (i = 0; i<n; i++)
       arr[i] = (int *)calloc(MAX_INDEX, sizeof(int));
```

[DS18]

**********DFS BFS 코드 (♥중♥요♥)

```
void dfs(int v) {

   adjPointer w;
   visited[v] = 1;
   printf("%d ", v);
   for (w = adjLists[v][0]; w; w = w->link)
       if (!visited[w->data])
           dfs(w->data);
}

int q[MAX] = { 0, };
int front=-1,rear = -1;
void bfs(int v) {
   adjPointer w;
   front = rear = -1;
   printf("%d ", v);
   visited2[v] = 1;
   q[++rear] = v;//add
   while (front!=rear) {
       v = q[++front];
       for (w = adjLists[v][0]; w; w = w->link) {
           if (!visited2[w->data]) {
               printf("%d ", w->data);
               q[++rear] = w->data;
               visited2[w->data] = 1;
           }
       }
   }
}
```

[DS19] 크루스칼

-> 사이클 검사 어려움

```
void run(int i, int j,int n,int count) {
    int max,min,x;
    if (cycle[i] == 0 && cycle[j] == 0) {
        cycle[i] = count; cycle[j] = count;
    }
    else {
        max = cycle[i] > cycle[j] ? cycle[i] : cycle[j]; //큰 값
        min= cycle[i] > cycle[j] ? cycle[j] : cycle[i]; //작은 값
        for (x = 0; x < n; x++) {
             if (cycle[x] == min) cycle[x] = max;
        }
    }
}

//cycle이면 1
int isCycle(int i, int j) {
    if (cycle[i] == cycle[j]&&cycle[i]!=0) return 1;
    return 0;
}
for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (isCycle(i, j) == 0 && arr[i][j] != 0) {
                if (arr[i][j] < min) { min = arr[i][j]; mini = i; minj = j; }
            }}}

    printf("Selected Edges: ");
    run(mini, minj, n, count++);
    printf(" (%d, %d),", mini, minj);
    ban[mini][minj] = 1;
    ban[minj][mini] = 1;
    ans += arr[mini][minj];
    cnt++;
```

[DS20] AOV 위상정렬

코드 매우 어려움

```
void topSort(hdnodes graph[], int n) {
   int i, j, k, top;
   nodePointer ptr;

   top = -1;

   for (i = 0; i< n; i++)
      if (!graph[i].count) {
         graph[i].count = top;
         top = i;
      }

   for (i = 0;i<n; i++)
      if (top == -1) {
         fprintf(stderr, "\nNetwork has a cycle. Sort terminated. \n");
         exit(1);
      }
      else {
         j = top;
         top = graph[top].count;
         printf("%d, ", j);
         for (ptr = graph[j].link; ptr; ptr = ptr->link) {
            k = ptr->vertex;
            graph[k].count--;
            if (!graph[k].count) {
                graph[k].count = top;
                top = k;
            }
        }
    }
}



    topSort(graph, n);
```

[DS21] QS ,HS 코드 잘알아야함

```
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void quickSort_a(int a[], int b[], int left, int right)
{
    int pivot, i, j, cnt = 0;
    if (left < right) {
        i = left; j = right + 1;
        pivot = a[left];
        do {
            do i++; while (a[i] < pivot);
            do j--; while (a[j] > pivot);
            if (i < j) {
                swap(&a[i], &a[j]);
                swap(&b[i], &b[j]);
            }
        } while (i < j);
        swap(&a[left], &a[j]);
        swap(&b[left], &b[j]);
        quickSort_a(a, b, left, j - 1);
        quickSort_a(a, b, j + 1, right);
    }
}

void adjust(int a[], int b[], int root, int n) {
    int child, rootkey, temp, temp2;
    temp = a[root];
    temp2 = b[root];
    rootkey = a[root];
    child = 2 * root;
    while (child <= n) {
        if ((child < n) && (a[child] < a[child + 1]))
            child++;
        if (rootkey > a[child]) break;
        else {
            a[child / 2] = a[child];
            b[child / 2] = b[child];
            child *= 2;
        }
    }
    a[child / 2] = temp;
    b[child / 2] = temp2;

}
void heapSort(int a[], int b[], int n) {
    int i, j;
    for (i = n / 2; i > 0; i--) {
        adjust(a, b, i, n);
    }
    for (i = n - 1; i > 0; i--) {
        swap(&a[1], &a[i + 1]);
        swap(&b[1], &b[i + 1]);
        adjust(a, b, 1, i);
    }
}
    quickSort_a(arr1_qs1, arr2_qs1, 0, n - 1);
    heapSort(arr1_hs1, arr2_hs1, n);
```