자료구조 프로그래밍 과제3 제출

호크마교양대학 장예서

정렬에 사용한 데이터 설명

- 2만 개의 정수 데이터로 정렬 수행. 정수 데이터의 선정은 rand() 함수를 이용하여 난수 2만 개를 생성하여 사용함.

1. Selection Sort 선택 정렬

선택 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SWAP(x, y, t) ( (t)=(x), (x)=(y), (y)=(t) )
#define MAX_SIZE 200000

void selection_sort(int list[], int n) {
	int i, j, least, temp;
	for (i = 0; i < n - 1; i++) {
		least = i;
		for (j = i + 1; j < n; j++)
			if (list[j] < list[least]) least = j;
		SWAP(list[i], list[least], temp);
	}
}

int main(void) {
	clock_t start, end;
	start = clock();
	int i;
	int n = MAX_SIZE;
	int list[MAX_SIZE];
	srand(time(NULL));
	for (i = 0; i < n; i++) list[i] = rand();
	selection_sort(list, n);
	for (i = 0; i < n; i++) printf("%d ", list[i]);
	end = clock();
	double result = (double)(end - start)/CLOCKS_PER_SEC;
	printf("₩n₩n");
	printf("--------- 선택 정렬 -------- ₩n소요 시간: %lf₩n", result);
	return 0;
}
```

선택 정렬 실행 결과 화면:

```
32762 32763 32763 32763 32763 32763 32763 32763 32764 32764 32764 32764 32764 32764 32764 32765 32765 32765 32765 32765
32765 32766 32766 32766 32766 32766 32767 32767 32767 32767 32767 32767 32767 32767 32767 32767

--------- 선택 정렬 --------
소요 시간: 94.889000
```

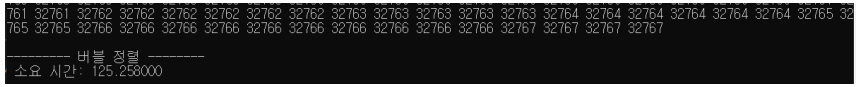
2. Bubble Sort 버블 정렬

버블 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SWAP(x, y, t) ( (t)=(x), (x)=(y), (y)=(t) )
#define MAX_SIZE 200000

void bubble_sort(int list[], int n) {
	int i, j, temp;
	for (i = n - 1; i > 0; i--) {
		for (j = 0; j < i; j++)
			if (list[j] > list[j + 1])
				SWAP(list[j], list[j + 1], temp);
	}
}

int main(void) {
	clock_t start, end;
	start = clock();
	int i;
	int n = MAX_SIZE;
	int list[MAX_SIZE];
	srand(time(NULL));
	for (i = 0; i < n; i++) list[i] = rand();
	bubble_sort(list, n);
	for (i = 0; i < n; i++) printf("%d ", list[i]);
	end = clock();
	double result = (double)(end - start) / CLOCKS_PER_SEC;
	printf("₩n₩n");
```

```
        printf("--------- 버블 정렬 -------- ₩n 소요 시간: %lf₩n", result);
        return 0;
}
```

버블 정렬 실행 결과 화면:

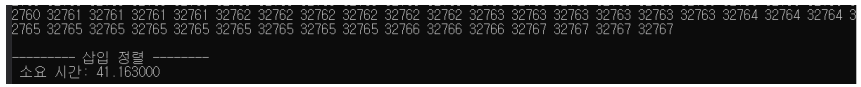


3. Insertion Sort 삽입 정렬

삽입 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000

void insertion_sort(int list[], int n) {
        int i, j, key;
        for (i = 1; i < n; i++) {
                key = list[i];
                for (j = i - 1; j >= 0 && list[j] > key; j--)
                        list[j + 1] = list[j];
                list[j + 1] = key;
        }
}

int main(void) {
        clock_t start, end;
        start = clock();
        int i;
        int n = MAX_SIZE;
        int list[MAX_SIZE];
        srand(time(NULL));
        for (i = 0; i < n; i++) list[i] = rand();
        insertion_sort(list, n);
        for (i = 0; i < n; i++) printf("%d ", list[i]);
        end = clock();
        double result = (double)(end - start) / CLOCKS_PER_SEC;
        printf("₩n₩n");
        printf("--------- 삽입 정렬 -------- ₩n 소요 시간: %lf₩n", result);
        return 0;
}
```

삽입 정렬 실행 결과 화면:



4. Shell Sort 쉘 정렬

쉘 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000

inc_insertion_sort(int list[], int first, int last, int gap) {
        int i, j, key;
        for (i = first + gap; i <= last; i = i + gap) {
                key = list[i];
                for (j = i - gap; j >= first && list[j] > key; j = j - gap)
                        list[j + gap] = list[j];
                list[j + gap] = key;
        }
}

void shell_sort(int list[], int n) {
        int i, gap;
        for (gap = n / 2; gap > 0; gap = gap / 2) {
                if ((gap % 2) == 0) gap++;
                        for (i = 0; i < gap; i++)
                                inc_insertion_sort(list, i, n - 1, gap);
        }
}

int main(void) {
        clock_t start, end;
```

```
	start = clock();
	int i;
	int n = MAX_SIZE;
	int list[MAX_SIZE];
	srand(time(NULL));
	for (i = 0; i < n; i++) list[i] = rand();
	shell_sort(list, n);
	for (i = 0; i < n; i++) printf("%d ", list[i]);
	end = clock();
	double result = (double)(end - start) / CLOCKS_PER_SEC;
	printf("₩n₩n");
	printf("--------- 쉘 정렬 -------- ₩n 소요 시간: %lf₩n", result);
	return 0;
}
```

쉘 정렬 실행 결과 화면:

```
62 32762 32762 32762 32763 32763 32763 32763 32763 32764 32764 32764 32764 32764 32764 32764 32764 32765 32765 327
65 32765 32765 32765 32766 32766 32766 32766 32766 32766 32766 32767 32767 32767 32767 32767 32767 32767

--------- 쉘 정렬 --------
소요 시간: 18.528000
```

5. Heap Sort 히프 정렬

히프 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000

typedef struct {
	int key;
} element;
typedef struct {
	element heap[MAX_SIZE+5];
	int heap_size;
} HeapType;

HeapType* create() {
	return (HeapType*)malloc(sizeof(HeapType));
}

void init(HeapType* h) {
	h->heap_size = 0;
}

void insert_max_heap(HeapType* h, element item) {
	int i;
	i = ++(h->heap_size);
	while ((i != 1) && (item.key > h->heap[i / 2].key)) {
		h->heap[i] = h->heap[i / 2];
		i /= 2;
	}
	h->heap[i] = item;
}

element delete_max_heap(HeapType* h) {
	int parent, child;
	element item, temp;
	item = h->heap[1];
	temp = h->heap[(h->heap_size)--];
	parent = 1;
	child = 2;
	while (child <= h->heap_size) {
		if ((child < h->heap_size) &&
			(h->heap[child].key) < h->heap[child + 1].key)
			child++;
		if (temp.key >= h->heap[child].key) break;
		h->heap[parent] = h->heap[child];
		parent = child;
		child *= 2;
	}
	h->heap[parent] = temp;
	return item;
}

void heap_sort(element a[], int n) {
	int i;
	HeapType* h;
	h = create();
	init(h);
	for (i = 0; i < n; i++) {
		insert_max_heap(h, a[i]);
```

```
        }
        for (i = (n - 1); i >= 0; i--) {
                a[i] = delete_max_heap(h);
        }
        free(h);
}

int main(void) {
        clock_t start, end;
        start = clock();
        int i;
        int n = MAX_SIZE;
        element list[MAX_SIZE];
        srand(time(NULL));
        for (i = 0; i < n; i++)
                list[i].key = rand();
        heap_sort(list, MAX_SIZE);
        for (i = 0; i < n; i++) printf("%d ", list[i]);

        end = clock();
        double result = (double)(end - start) / CLOCKS_PER_SEC;
        printf("₩n₩n");
        printf("--------- 히프 정렬 -------- ₩n 소요 시간: %lf₩n", result);
        return 0;
}
```

히프 정렬 실행 결과 화면:

```
63 32763 32763 32763 32763 32763 32764 32764 32764 32764 32764 32764 32764 32764 32765 32765 32765 32766 327
66 32766 32766 32766 32766 32767 32767 32767

--------- 히프 정렬 --------
소요 시간: 17.768000
```

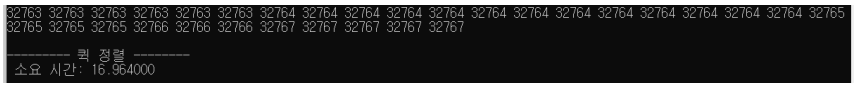
6. Quick Sort 퀵 정렬

퀵 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000
#define SWAP(x, y, t) ( (t)=(x), (x)=(y), (y)=(t) )

int partition(int list[], int left, int right) {
        int pivot, temp;
        int low, high;
        low = left;
        high = right + 1;
        pivot = list[left];
        do {
                do
                        low++;
                while (low <= right && list[low] < pivot);
                do
                        high--;
                while (high >= left && list[high] > pivot);
                if (low < high) SWAP(list[low], list[high], temp);
        } while (low < high);
        SWAP(list[left], list[high], temp);
        return high;
}

void quick_sort(int list[], int left, int right) {
        if (left < right) {
                int q = partition(list, left, right);
                quick_sort(list, left, q - 1);
                quick_sort(list, q + 1, right);
        }
}

int main(void) {
        clock_t start, end;
        start = clock();

        int i;
        int n = MAX_SIZE;
        int list[MAX_SIZE];
        srand(time(NULL));
        for (i = 0; i < n; i++) list[i] = rand();
        quick_sort(list, 0, n - 1);
        for (i = 0; i < n; i++) printf("%d ", list[i]);

        end = clock();
        double result = (double)(end - start) / CLOCKS_PER_SEC;
        printf("₩n₩n");
```

```
        printf("--------- 퀵 정렬 -------- ₩n 소요 시간: %lf₩n", result);
        return 0;
}
```

퀵 정렬 실행 결과 화면:

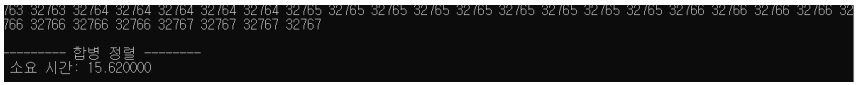


7. Merge Sort 합병 정렬

합병 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000

int sorted[MAX_SIZE];
void merge(int list[], int left, int mid, int right) {
        int i, j, k, l;
        i = left; j = mid + 1; k = left;
        while (i <= mid && j <= right) {
                if (list[i] <= list[j]) sorted[k++] = list[i++];
                else sorted[k++] = list[j++];
        }
        if (i > mid)
                for (l = j; l <= right; l++)
                        sorted[k++] = list[l];
        else
                for (l = i; l <= mid; l++)
                        sorted[k++] = list[l];
        for (l = left; l <= right; l++)
                list[l] = sorted[l];
}

void merge_sort(int list[], int left, int right) {
        int mid;
        if (left < right) {
                mid = (left + right) / 2;
                merge_sort(list, left, mid);
                merge_sort(list, mid + 1, right);
                merge(list, left, mid, right);
        }
}


int main(void) {
        clock_t start, end;
        start = clock();

        int i;
        int n = MAX_SIZE;
        int list[MAX_SIZE];
        srand(time(NULL));
        for (i = 0; i < n; i++) list[i] = rand();
        merge_sort(list, 0, n - 1);
        for (i = 0; i < n; i++) printf("%d ", list[i]);

        end = clock();
        double result = (double)(end - start) / CLOCKS_PER_SEC;
        printf("₩n₩n");
        printf("--------- 합병 정렬 -------- ₩n 소요 시간: %lf₩n", result);
        return 0;
}
```

합병 정렬 실행 결과 화면:



8. Radix Sort 래딕스 정렬

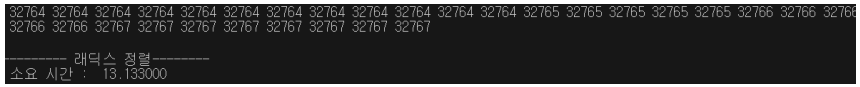
래딕스 정렬 소스 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 200000
```

```c
typedef int element;
typedef struct QueueNode {
	element data;
	struct QueueNode* link;
} QueueNode;
typedef struct {
	QueueNode* front, * rear;
	int count;
} LinkedQueueType;

void InitQueue(LinkedQueueType* queue) {
	queue->front = queue->rear = NULL;
	queue->count = 0;
}

int IsEmpty(LinkedQueueType* queue) {
	return queue->count == 0;
}

void enqueue(LinkedQueueType* q, element data){
	QueueNode* temp = (QueueNode*)malloc(sizeof(QueueNode));
	temp->data = data;
	temp->link = NULL;
	if (IsEmpty(q)) {
		q->front = temp;
		q->rear = temp;
	}
	else {
		q->rear->link = temp;
		q->rear = temp;
	}
}

element dequeue(LinkedQueueType* q){
	QueueNode* temp = q->front;
	element data;
	if (IsEmpty(q)) {
		fprintf(stderr, "스택이 비어있음₩n");
		exit(1);
	}
	else {
		data = temp->data;
		q->front = q->front->link;
		if (q->front == NULL)
			q->rear = NULL;
		free(temp);
		return data;
	}
}

#define BUCKETS 10
#define DIGITS 6

void radix_sort(int list[], int n){
	int i, b, d, factor = 1;
	LinkedQueueType queues[BUCKETS];
	for (b = 0; b < BUCKETS; b++) InitQueue(&queues[b]);
	for (d = 0; d < DIGITS; d++) {
		for (i = 0; i < n; i++) {
			enqueue(&queues[(list[i] / factor)%10], list[i]);
		}
		for (b = i = 0; b < BUCKETS; b++)
			while( !IsEmpty(&queues[b]) )
				list[i++] = dequeue(&queues[b]);
		factor *= 10;
	}
}

int main(void) {
	clock_t start, end;
	start = clock();
	int list[MAX_SIZE];
	srand(time(NULL));
	for (int i = 0; i < MAX_SIZE; i++)
		list[i] = rand();
	radix_sort(list, MAX_SIZE);
	for (int i = 0; i < MAX_SIZE; i++)
		printf("%d ", list[i]);
	printf("₩n");
	end = clock();
	double result = (double)(end - start) / CLOCKS_PER_SEC;
	printf("₩n₩n");
```

```
        printf("--------- 래딕스 정렬-------- ₩n 소요 시간 : % lf₩n", result);
        return 0;
}
```

래딕스 정렬 실행 결과 화면:



결과 설명

각 정렬이 정수 데이터 2만 개를 작은 순에서 큰 순으로 정렬하는 데에 소요한 시간은 다음과 같다.

정렬명	소요 시간 (단위: 초)
선택 정렬	94.889000
버블 정렬	125.258000
삽입 정렬	41.163000
쉘 정렬	18.528000
히프 정렬	17.768000
퀵 정렬	16.964000
합병 정렬	15.620000
래딕스 정렬	13.133000

따라서 정수 데이터 2만 개를 정렬하는 과정에서 정렬 알고리즘의 성능은,

래딕스 정렬 >= 합병 정렬 >= 퀵 정렬 >= 히프 정렬 >= 쉘 정렬 >>> 삽입 정렬 > 선택 정렬 > 버블 정렬

이다.