알고리즘 과제 1

제출 마감일 : 2016.10.16

담당교수 : 최지웅 교수님
소속학부 : 컴퓨터학부
학번 : 20142577
이름 : 홍상원

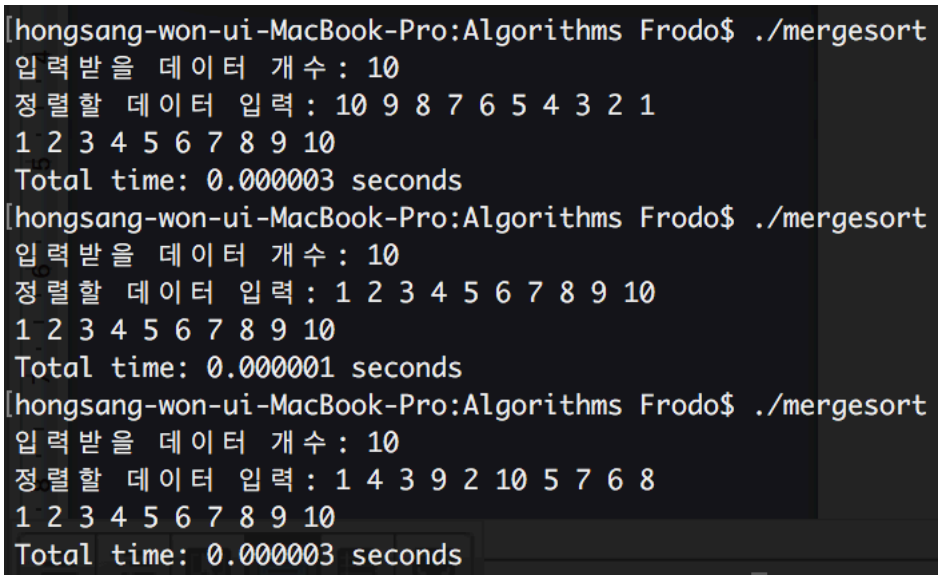
1. mergesort
1.1. mergesort 구현 : MergeSort.c

```
//비내림차순으로 정렬
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

typedef int keytype;
typedef int index;

void merge(int h, int m, const keytype U[], const keytype V[], keytype S[]) {
	index i = 0, j = 0, k = 0;

	//각각 정렬된 배열 U 와 배열 V 의 원소들을 비교하여 배열 S 에 정렬하면서 배열 U 와 배열 V 를 배열 S 에 합병한다.
	while(i < h && j < m) {
		if(U[i] < V[j]) {
			S[k] = U[i];
			i++;
		}
		else {
			S[k] = V[j];
			j++;
		}
		k++;
	}

	//비교 후 배열 U 또는 배열 V 에 배열 S 에 저장하지 않고  남아있는 원소들을 모두 배열 S 에 복사한다.
	if(i >= h) {
		for(index p = j; p < m; p++, k++)
			S[k] = V[p];
	}

	if(j >= m) {
		for(index p = i; p < h; p++, k++)
			S[k] = U[p];
	}
}

void merge_sort(int n, keytype S[]) {
```

```
	if(n > 1) {
		const  int h = n / 2, m = n - h;
		keytype U[h], V[m];
		//배열 S 를 배열 U 와 배열 V 에 나누어 저장한다.
		for(index i = 0; i < h; i++)
			U[i] = S[i];
		for(int i = 0; i < m; i++)
			V[i] = S[h + i];

		merge_sort(h, U);//배열 U mergesort 진행
		merge_sort(m, V);//배열 V mergesort 진행
		merge(h, m, U, V, S);//각각 정렬된 배열 U 와 배열 V 를 배열 S 에 합병
정렬하면서 저장 한다.
	}
}

int main(void) {
	keytype *S;
	int N;
	struct timeval begin, end;//merge_sort 실행시간 측정
	double time;

	printf("입력받을 데이터 개수: ");
	scanf("%d", &N);

	S = (keytype *)malloc(sizeof(keytype) * N);
	printf("정렬할 데이터 입력: ");
	for(index i = 0; i < N; i++)
		scanf("%d", &S[i]);

	gettimeofday(&begin, NULL);
	merge_sort(N, S);
	gettimeofday(&end, NULL);

	time = (double)(end.tv_usec - begin.tv_usec) / 1000000 +
(double)(end.tv_sec - begin.tv_sec);
	for(index i = 0; i < N; i++)
		printf("%d ", S[i]);
	printf("₩n");
	printf("Total time: %lf seconds₩n", time);
	free(S);
	return 0;
```

}

1.2. mergesort 실행결과



2. quicksort

2.1. quicksort 구현 : QuickSort.c

```
//비내림차순으로 정렬
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

typedef int index;
typedef int keytype;

//교환함수
void swap(keytype *itemA, keytype *itemB) {
        keytype temp = *itemA;
        *itemA = *itemB;
        *itemB = temp;
}

void partition(index low, index high, index *pivotpoint, keytype S[]) {
        index i, j;
        keytype pivotitem;//pivot 으로 삼을 값 저장
```

```
	pivotitem = S[low];
	j = low;//j 로 pivotpoint 찾기
	for(i = low + 1; i <= high; i++) {
		//pivotitem 보다 작은 값은 pivotitem 왼쪽으로 이동
		if(S[i] < pivotitem) {
			j++;
			swap(&S[i], &S[j]);
		}
	}
	*pivotpoint = j;
	swap(&S[low], &S[*pivotpoint]);
}

void quicksort(index low, index high, keytype S[]) {
	index pivotpoint;//pivot index 저장

	if(high > low) {
		partition(low, high, &pivotpoint, S);//pivotpoint 를 중심으로 나눈다.
		quicksort(low, pivotpoint - 1, S);//pivotpoint 왼쪽 정렬
		quicksort(pivotpoint + 1, high, S);//pivotpoint 오른쪽 정렬
	}
}
int main(void) {
	keytype *S;
	int N;
	struct timeval begin, end;//quicksort 실행시간 측정
	double time;

	printf("입력할 데이터 개수: ");
	scanf("%d", &N);

	S = (keytype *)malloc(sizeof(keytype) * N);
	printf("정렬할 데이터 입력: ");
	for(index i = 0; i < N; i++)
		scanf("%d", &S[i]);

	gettimeofday(&begin, NULL);
	quicksort(0, N - 1, S);
	gettimeofday(&end, NULL);
```

```
	time = (double)(end.tv_usec - begin.tv_usec) / 1000000 + (double)(end.tv_sec - begin.tv_sec);

	for(index i = 0; i < N; i++)
		printf("%d ", S[i]);
	printf("₩n");
	printf("Total time: %lf seconds₩n", time);

	free(S);
	return 0;
}
```

2.2. quicksort 실행결과

```
hongsang-won-ui-MacBook-Pro:Algorithms Frodo$ ./quicksort
입력할 데이터 개수 : 10
정렬할 데이터 입력 : 8 4 1 7 2 9 3 10 5 11
1 2 3 4 5 7 8 9 10 11
Total time: 0.000001 seconds
hongsang-won-ui-MacBook-Pro:Algorithms Frodo$ ./quicksort
입력할 데이터 개수 : 15
정렬할 데이터 입력 : 10 8 13 51 23 53 33 21 5 6 1 22 11 7 4
1 4 5 6 7 8 10 11 13 21 22 23 33 51 53
Total time: 0.000002 seconds
hongsang-won-ui-MacBook-Pro:Algorithms Frodo$ ./quicksort
입력할 데이터 개수 : 10
정렬할 데이터 입력 : 1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
Total time: 0.000001 seconds
hongsang-won-ui-MacBook-Pro:Algorithms Frodo$ ./quicksort
```