[R E P O R T]

정보통신공학전공

200301582

김성태

국립공주대학교

KONGJU NATIONAL UNIVERSITY

QuickSort 알고리즘 구현 과제 보고서

1. 과제의 목적

① QuickSort 알고리즘의 동작방식을 이해한다.

② QuickSort 알고리즘의 시간복잡도를 계산한다.

③ Stack의 구조를 이해한다.

④ QuickSort 알고리즘의 의사코드를 프로그래밍 언어로 구현한다.

2. 필요 환경

① GCC

② Linux

3. 과제 수행 내역

① QuickSort 알고리즘의 동작 방식 이해

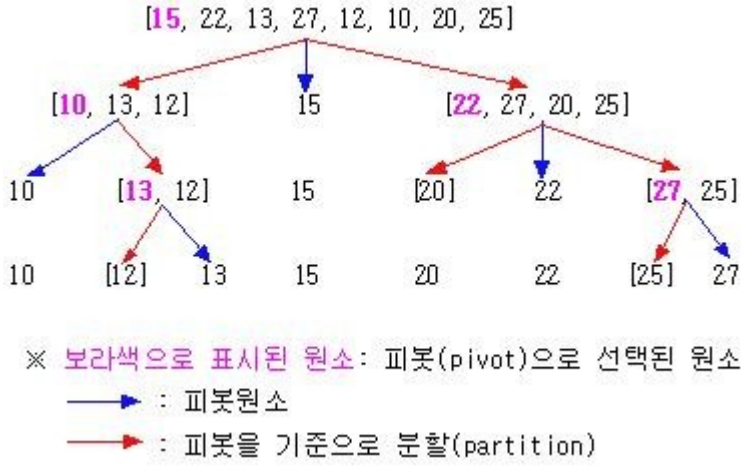
② QuickSort 알고리즘의 시간복잡도 계산

③ Stack의 구조 이해

④ QuickSort 알고리즘의 구현

4. 과제 수행

① QuickSort 알고리즘의 동작 방식 이해



퀵 정렬에서는 피봇(pivot) 원소를 기준으로 하여 리스트를 분할시킨다.

리스트 내의 어떠한 원소도 피봇이 될 수 있지만, 편의상 주어진 리스트의 첫 번째 원소를 피봇으로 한다.

그런 후, 우리가 정렬하고자 하는 리스트를 ①피봇보다 작은 값을 갖는 원소들로 구성된 리스트와 ②피봇보다 값이 큰 원소들로 구성된 리스트로 분할시킨다.
분할 과정을 계속하여 충분히 작은 리스트(1개의 원소로 이루어진 리스트)가 얻어질 때까지, 순환적으로 적용함으로써 전체 리스트를 정렬하는 방법이다.
이 방법을 사용하면, 합병 정렬(merge sort)처럼 합병(merge)의 과정이 필요없다는 장점이 있다.

② Quick Sort 알고리즘의 시간복잡도 계산

<최악의 경우>
최악의 경우는 배열이 이미 정렬되어 있는 경우이다. 분할과정(partition)에서 연산횟수 T(n)에 대하여 다음의 식이 성립한다.

$T(n) = T(n-1) + n - 1 \quad , n > 0$

$T(n) = 0 \quad , n = 0$

위의 식에서...

$T(n-1)$: 피봇 원소의 오른쪽에 위치한 부분배열을 정렬하는데 소요되는 시간.

$n-1$: 분할(partition) 과정에서 소요되는 시간

관계식을 정리하면...최악의 경우 시간 복잡도는

$T(n) = \{n(n-1)\} / 2 = O(n^2)$

이 된다.

<평균의 경우>
다음은 평균적인 시간복잡도를 구하는 관계식이다.

$$C_A(n) = n + 1 + \frac{1}{n} \sum_{1 \le k \le n} [C_A(k-1) + C_A(n-k)]$$

- 피봇보다 큰 원소들을 정렬하는데 소요되는 시간
- 피봇보다 작은 원소들을 정렬하는 소요되는 시간
- 피봇이 정렬된 배열에서 i번째 원소일 확률
- 분할과정에서의 비교횟수

처음 $(n+1)$은 원소의 비교횟수를 나타낸다. $1/n$ 은 분할과정 후 피봇의 위치가 i번째일 확률이다. $C_A(k-1)$과 $C_A(n-k)$ 는 각 부분배열에 대해 순환적인 분할 과정을 수행하는데 소요되는 시간을 의미한다.
....계산하면 평균의 시간복잡도는 O(n lg n) 의 시간 복잡도가 나온다.

③ Stack 구조의 이해

스택은 제한적으로 접근될 수 있는 리스트(list-structure)구조이다. 그 접근방법은 항상 리스트의 끝인 TOP 에서만 일어난다.Pushdown list 라고도 한다.스택은 한쪽 끝에서만 자료를 넣거나 뺄 수 있는 선형 구조로 되어있다.자료를 넣는 것을 '밀어넣는다' 하여 **푸시**(push)라고 한고 반대로 넣어둔 자료를 꺼내는 것을 **팝**(pop)이라고 하는데, 이 때 꺼내지는 자료는 가장 최근에 보관한 자료부터 나오게 된다. 이처럼 나중에 넣은 값이 먼저 나오는 후입선출(Last In First Out)구조로 되어 **LIFO** 로도 부른다.

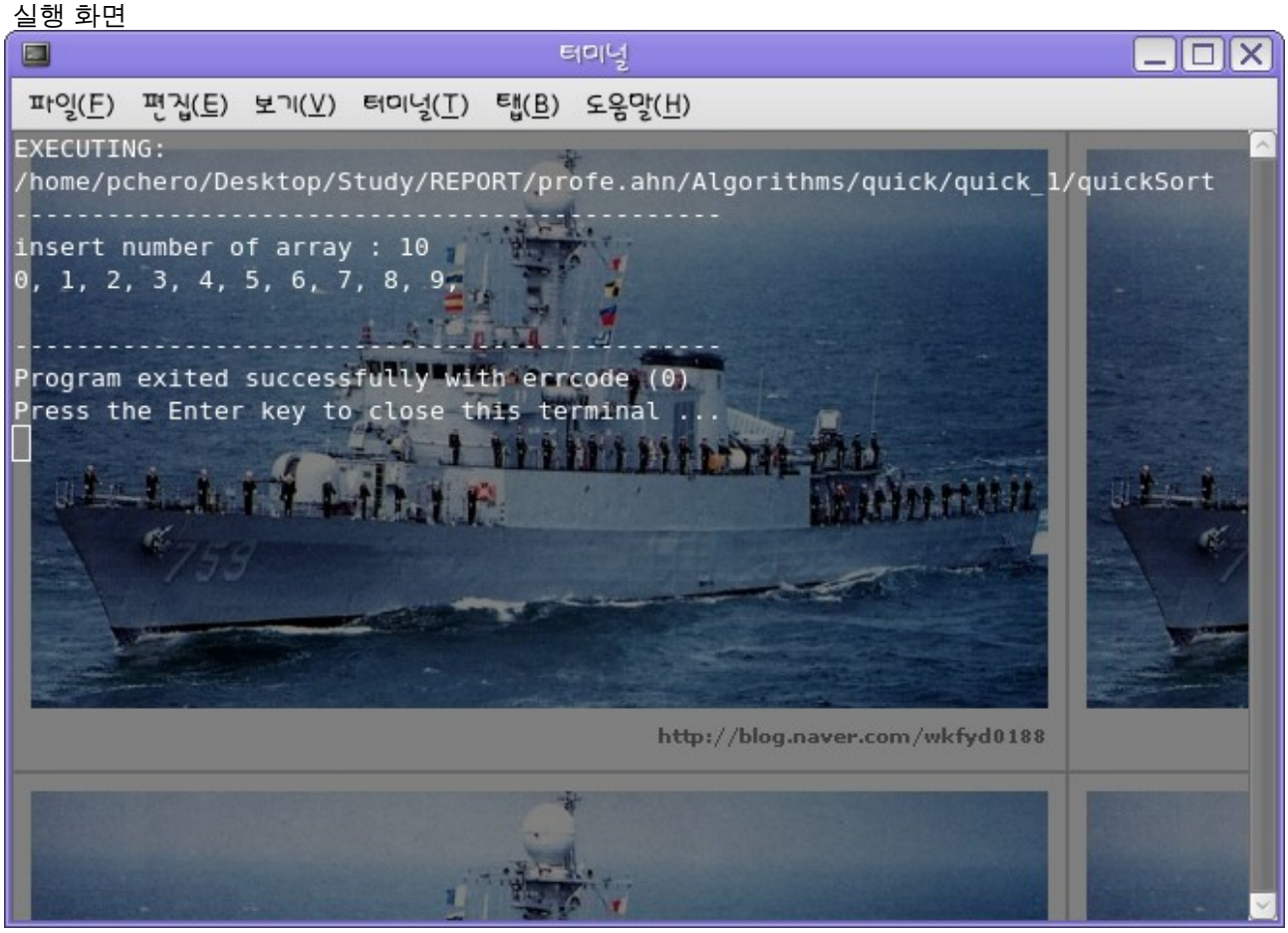
④ Quick Sort 알고리즘 구현

```
/*************************************************************************
 *        quickSort.c
 *
 *  Tue Oct 16 19:50:06 2007
 *  Copyright  2007  pchero
 *  pchero21@gmail.com
 *************************************************************************/

/*
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 */

#include <stdio.h>
#include <stdlib.h>

void quickSort(int low, int high);
void partition(int low, int high, int *pivotpoint);
void swap(int *a, int *b);

int *S;

int main(int argc, int **argv)
{

        int i, j;
        int arr_size;

        printf("insert number of array : ");
```

```
	scanf("%d", &arr_size);

	S = (int*)malloc(sizeof(int) * arr_size);

	// 난수 발생.
	j = arr_size;
	for(i = 0; i < arr_size; i++)
		S[i] = i;
	for(i = 0; i < (arr_size -1); i++)
		swap(&S[rand() % j], &S[--j]);

	quickSort(0, (arr_size - 1));

	for(i = 0; i < arr_size; i++)
		printf("%d\n", S[i]);

	free(S);
	return 0;
}

void quickSort(int low, int high)
{
	int pivotpoint;

	if(high > low) {
		partition(low, high, &pivotpoint);
		quickSort(low, pivotpoint - 1);
		quickSort(pivotpoint + 1, high);
	}
}

void partition(int low, int high, int *pivotpoint)
{
	int i, j;
	int pivotitem;

	pivotitem = S[low];
	j = low;
	for(i = low + 1; i <= high; i++) {
		if(S[i] < pivotitem) {
```

```
                        j++;
                        swap(&S[i], &S[j]);
                }
        }
        *pivotpoint = j;
        swap(&S[low], &S[*pivotpoint]);
}

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

실행 화면



5. 느낀점

평소 자료구조에 대해 알고는 있었으나....Stack을 이용한 Quick Sort 구현을 하지 못했습니다. 예제 소스

를 구하여 분석을 시도 하였으나, 사용된 알고리즘을 이해하지 못하여 구현하지 못했습니다. 하지만, 이번 과제로 인하여 자료구조에대해 다시한번 생각해 보는 계기가 되었습니다.

6. 참고 사이트

① http://kldp.org 한국 리눅스 문서 프로젝트
② http://phpshool.com 한국 PHP 유저 포럼
③ http://debianusers.org 한국 데비안 유저 포럼
④ http://google.com/codesearch 구글 소스 검색