

| Oral Session IX : Big Data, Smart Energy ICT, Smart Information

좌장 : 신춘성 (전남대)

업종별 부채 예측 모델 개발 : 코로나 19 상황에서

김양석, 노미진, 김차미, 손승연, 조유진 (계명대학교) 114

녹조 발생 예측 AI모델 개발 연구

송수영, 송유선, 이유진, 홍경석, 김남호, 최광미 (호남대학교), 정희자(휴넷가이아) 116

Non-IID 환경에서 연합 학습 기반 전기 수요 예측

염성웅, Kolekar Shivani Sanjay, 조현준, 김경백 (전남대학교) 118

유사 서비스 함수를 위한 코드 모듈들의 구조 내 저전력 연구

윤예동, 문소영, 김영철 (홍익대학교) 120

복잡한 코드의 간결화를 통한 성능 및 저전력 개선

조재형, 문소영, 김영철 (홍익대학교) 123

CCTV 영상처리를 통한 화재감지기 오탐 개선에 관한 연구

황은호, 김남호 (호남대학교) 126

Knowledge Graph 확장을 위한 딥러닝 기반 관계 추출

최준호, 김형주 (조선대학교) 209

농경지 침수 분석을 위한 SWMM 모형의 적용성 검토

김규민, 원다윗, 양승원 (우석대학교) 211

공간정보 기반 농경지 침수피해의 선제적 대응을 위한 기초자료 구축

박석우, 양승원, 나인호 (군산대학교) 213

SWMM 해석 기반 공간분석 농경지 침수의 선제적 대응 연구

손성민, 김형진 (전북대학교) 215

색 추출 기법을 접목한 아트 플랫폼의 기대효과

유세빈, 황시준, 박남홍 (조선대학교) 217

알츠하이머병에 라지 스케일 네트워크의 연결 패턴 분석

라마라매쉬쿠마, 권구락 (조선대학교) 219

클라우드 컴퓨팅에서의 장애 허용 기법 분석

조만규, 이재환, 김찬수, 박상오 (중앙대학교) 222

기능점수 기반 정교한 비용 예측 추출을 위한 요구사항 스펙 구조화

문소영, 김영철 (홍익대학교) 224

신재생에너지 스마트팜 환경 기반 에너지 사용량 예측

임종현, 장경민, 오한별, 이명배, 신창선, 박장우, 조용윤 (순천대학교) 226

Firebase 클라우드 메시징을 활용한 스마트 헬스케어 플랫폼

남재경, 최민 (충북대학교), 김성준(중원대학교) 228

수경재배 양액관리를 위한 스마트 단말 모니터링 및 제어 시스템 구현

오한별, 이명배, 박장우, 조용윤, 신창선 (순천대학교) 230

데이터 분석 기반의 파프리카 온실 환경 예측에 대한 연구

장경민, 이명배, 조용윤, 신창선, 박장우 (순천대학교) 232

딥러닝 모델을 이용한 발전량 예측 방법

김지인, 이건우, 권구락 (조선대학교) 234

AMI 시스템에서 수집 시간 단축을 위한 기법 연구

나채훈, 김정인, 윤범식, 강향숙, 김판구 (조선대학교) 236

복잡한 코드의 간결화를 통한 성능 및 저전력 개선

조재형, 문소영, 김영철

홍익대학교 소프트웨어융합학과 소프트웨어공학 연구실

e-mail : henrycho@mail.hongik.ac.kr, whit2@hongik.ac.kr, bob@hongik.ac.kr

Improving Performance and Lower power consumption with Simplification of Code Complexity

Jae Hyeong Cho, So Young Moon, R. Young Chul Kim

SE Lab., Dept. of Software and Communications Engineering (Eng.), Hongik
University

요 약

최근 다양한 분야에서 기존 소프트웨어는 고성능을 유지하면서 제한된 메모리와 전력 등과 같이 제한된 환경에서도 안정적으로 작동이 가능하게 하는 소프트웨어 신뢰성이 요구되고 있다. 높은 사양의 하드웨어에서 성능이 높은 소프트웨어가 작동함으로써 불필요한 소비 전력이 증가하고 있다. 그러므로 코드 복잡도를 낮추는 것이 얼마나 성능과 저전력 소비를 줄이는 연구가 필요하다. 이를 위해 유사 서비스 코드들의 McCabe 복잡도와 알고리즘의 시간/공간 효율성을 비교를 통해 상관관계를 찾고 싶다. 이를 통해 코드의 간결화가 잠재적인 오류 및 성능, 저전력 소비를 줄이는 것을 확인했다. 앞으로 소프트웨어 코드의 전력 측정을 통한 전력 소모 최소화 및 성능개선 방법을 제안 할 수 있을 것이다.

Keyword: McCabe, Complexity, LowPower

1. 서 론

사람이 데이터를 분석 가능한 양은 최소 수십에서 최대 수백 개의 데이터 정도지만 컴퓨터가 분석하는 데이터의 양은 최소 백만 개 단위이며 데이터베이스 같은 경우 이론상 무한한 양의 데이터를 분석한다. 이러한 환경에서 소프트웨어의 전력, 메모리 등에서 안정적인 작동을 위해서는 소비전력이 중요하다. 특히 초소형기거나 무인기 에서도 최적의 값을 찾기 위해서 소팅 알고리즘이 사용된다. 이를 해결하기 위해, 알고리즘 관점에서 다양한 소팅 알고리즘의 실제 소모되는 전력을 측정해서 어떤 알고리즘 그 패턴의 전력량을 최소화시키기 위한 연구를 하였다.

2. 관련 연구

Vetro가 제안한 Energy Code Smell은 소모 전력을 감소시킬 수 있는 가능성이 높은 패턴이다[1]. 하지만 전력량의 소모가 증가되는 경우도 있어 소비 전력을 감소시키는 데 적합하다고 볼 수 없다. 기존 연구에서는 간단한 패턴에 대해서만 논하였다[2]. 코드의 전력을 측정한 도구로는 Keil uVision5보드와 IDE를 사용하여 측정하고 싶은 알고리즘을 입력한다. 입력한 알고리즘을 Build를 한 뒤 오류가 없으면 보드에 업로드를 한 뒤 소비전력과 분석과 디버깅이 가능한 arm사의 Ulink plus 측정 도구로 전력값과 그래프를 측정 할 수 있다[3]. 기존 논문에서는 전력의 소모를 최소화 하는 알고리즘을 입력 한 뒤 전력값에 개선이 보이는 것에 그쳤으면 이번 논문에서는 전력값에 수

행되는 시간 값을 곱하여 실질적인 전력량을 측정 하였다.

3. 전력량 측정하기 위한 소팅 알고리즘 3.1 측정 방법

```
int i, j, k, temp;
int arr[ARRNUM];
int num = 0;
EventStartA(0);
for ( i=ARRNUM-1; i>=0; i--) {
    arr[i] = num++;
}
for ( i = 0; i < ARRNUM; i++) {
    temp = arr[i];
    for ( j = i - 1; j >= 0 && arr[j] > temp; j--) {
        arr[j + 1] = arr[j];
    }
    arr[j + 1] = temp;
}
EventStopA(0);
```

(그림 1) 전력 측정 범위

그림 1은 실험에서 사용한 코드의 일부이다. 데이터 사이즈만큼 랜덤한 숫자 발생시켜 배열에 저장한다. 저장된 랜덤한 숫자들은 입력으로 사용된다. 그리고 정확도를 높이기 위해 이 실험을 각 50회씩 반복 수행 하여 산술 평균을 내었다. 본 논문에서는 이런 정렬 알고리즘 중 기본적인 버블, 삽입, 선택 알고리즘에 대하여 언급한다.

(표 1) 각 정렬 알고리즘 측정값

정렬 알고리즘 측정값	
Bubble sort	1 T(tot)=4.41ms Q(tot)=187.44μs T(avg)=4.41ms T(min)=4.41ms T(max)=4.41ms Stop: "main.c" (139) t=4.41ms Q=187.44μs I=42.54mA U=3.29V
Insertion sort	1 T(tot)=4.55ms Q(tot)=193.50μs T(avg)=4.55ms T(min)=4.55ms T(max)=4.55ms Stop: "main.c" (137) t=4.55ms Q=193.50μs I=42.53mA U=3.29V
Selection sort	1 T(tot)=4.41ms Q(tot)=187.62μs T(avg)=4.41ms T(min)=4.41ms T(max)=4.41ms Stop: "main.c" (143) t=4.41ms Q=187.62μs I=42.50mA U=3.29V

3.2 기본 소팅 알고리즘에 대한 실험 결과

표 1에서 나오는 값들은 각 정렬에서 1부터 100까지 100개의 랜덤한 숫자를 입력하여 코드에서 필요한 부분만 선택 하여 전력량을 구한 측정 표이다 각 줄의 의미는 측정을 한번 할 때 나오는 평균 값을 나타낸다. 전력량의 공식은 $E = I \times V \times T$ 로 T는 시간, I는 전류량, V는 전압이다. 예시로 표 1에서 bubble sort의 경우를 계산해 보면 $I=42.54\text{mA}$, $V=3.3\text{V}$, $T=4.41\text{ms}$ 인데 공식을 구하기 위해서는 T의 값을 I의 값과 같게 고쳐주어야 한다. 결과로 $42.5 \times 3.3 \times 0.44 = 61.7\mu\text{J}$ 가 나온다.

(표 2) 정렬 알고리즘 별 시간 및 공간 복잡도

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$

표 2는 정렬 알고리즘별 시간, 공간 복잡도이다. 세 알고리즘은 시간복잡도 측면에서 평균과 최악의 경우 같다. 최선의 경우 selection sort가 좋지 않음을 알 수 있다.

-Bubble sort 알고리즘: 버블 정렬은 서로 인접한 두 숫자를 비교하여 정렬하는 알고리즘이며 인접한 2개의 숫자를 비교하여 순서대로 서로 교환하는 알고리즘이다. 위에서 설명한 공식대로 랜덤한 숫자 100개, 200개, 300개 순으로 각각 87.6μJ, 188.4μJ, 307.8μJ 순으로 측정되었다.

-Insertion sort 알고리즘: 삽입 정렬은 배열의 모든 숫자들을 앞에서 부터 차례대로 앞서 정렬된 비교 부분과, 자신의 비교 할 위치를 찾아 교환함으로써 정렬을 하는 알고리즘이다. 마찬가지로 측정 결과는 81.6μJ, 166.9μJ, 285.3μJ로 측정 되었다.

-Selection sort 알고리즘: 선택 정렬은 해당 순서에 교환 할 숫자를 교환 할 위치는 이미 정해져 있고, 어떤 숫자를 교환 할 지 선택하는 알고리즘이다. 측정한 결과는 85.6μJ, 183.0μJ, 305.5μJ로 측정되었다.

(표 3) 정렬 알고리즘 별 전력량 측정 그래프

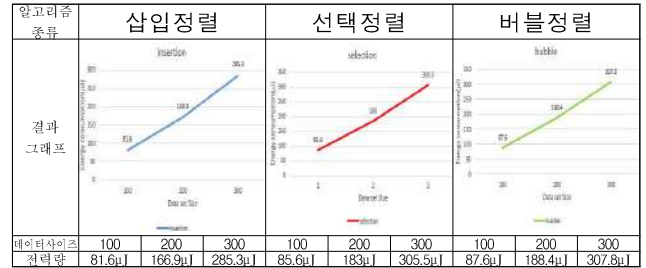


표3는 위에서 제시한 정렬 알고리즘들을 측정 후 그래프로 정리한 측정 표이다. 결과는 많은 차이가 나지 않지만 그 차이가 누적 되면 연산의 양이 많아지고 차이도 벌어질 것으로 예상된다.

3.3 순환 복잡도 및 #LOC의 상관관계 분석

(표 4) 각 정렬 알고리즘 순환복잡도(Cyclomatic Complexity)

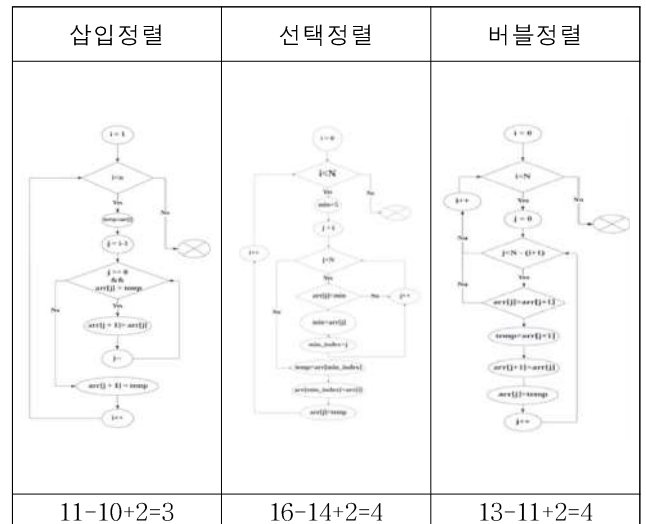


표4는 각 정렬 알고리즘의 핵심 코드에 대한 LOC와 전력량은 상관관계가 없었다. 표5는 위에서 제시한 알고리즘들을 순환복잡도로 계산하여 나열한 표이다. 순환복잡도가 낮게 수정하면 전력량이 낮아지는 것을 볼 수 있다.

4. 결론

본 연구는 정렬 알고리즘 코드를 이용하는 각종 프로그램들이 효율적으로 전력량을 줄일 수 있는 방법을 제안한다. 각 알고리즘 코드의 복잡도를 줄이면, 불필요하게 전력이 소모 되지 않도록 개선 할 수 있다. 향후 연구로는 유사 기능을 수행 하는 알고리즘에 대해서 복잡도를 낮추어 전력량을 줄이는 연구를 진행 할 계획이다.

ACKNOWLEDGMENT

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21사업의 지원(F21YY8102068)과 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원(No.2021R1I1A305040711, No.2021R1I1A1A01044060)을 받아 수행된 연구임.

참 고 문 헌

- [1] 안현식, 박보경, 김영철. 2020. 전력 소모 최소화를 통한 성능 개선의 코드 가시화 방법. 한국정보처리학회 학술대회논문집, 27(1): 375-376
- [2] 안현식, 박보경, 김영철, 김기두.(2020).성능 저하 식별을 통한 저전력 개선용 코드 가시화 방법.정보처리학회논문지. 컴퓨터 및 통신시스템,9(10),213-220.
- [3] 조재형, 윤예동, 장우성, 김영철. 2021 무인 자율 물체를 위한 SW코드의 저전력 소비 연구. 한국인터넷정보학회. 22(2): 25-26