알고리즘
프로그래밍 실습 3

인천대학교

알고리즘 | 지도교수 채진석

송병준 | 201701562

알고리즘

프로그래밍 실습 3

송병준 | 201701562
2019년 9월 26일

---

1. 과제

다음 알고리즘을 Python으로 구현

- isPerfect
- isPrime
- printNum

다음 정렬 알고리즘을 Python으로 구현, 실행시간 측정 및 비교

- cocktailShaker
- exchangeSort

2. 환경

운영체제: macOS 10.14.6 (Darwin Kernel Version 18.7.0 x86_64)
편집기: Atom 1.40.1
런타임: Python 2.7.10

3. 수행

3.1 알고리즘 구현

- **isPerfect**

완전수는 약수의 합이 자기 자신과 같은 수이다. 약수의 합을 모두 구해 해당 수와 비교하면 된다.

먼저 약수의 합을 구하는 함수 divisorSum을 다음과 같이 구현한다.

```
def divisorSum(n):
	if (n < 1): return 0
	return sum(map(lambda d: d if n % d == 0 else 0, range(1, n)))
```

그리고 isPerfect를 구현한다.

```
def isPerfect(n):
	return divisorSum(n) - n
```

- **isPrime**

소수는 약수가 1과 자기 자신뿐인 수이다. 약수의 합이 1인지 비교하면 된다.

구현에는 위에서 정의한 divisorSum을 사용한다.

```
def isPrime(n):
	return divisorSum(n) == 1
```

- **printNum**

1부터 주어진 숫자까지 출력하는데, 이때 각 줄의 길이는 2의 제곱수를 따른다.

현재 출력하는 숫자가 해당 줄에서 몇 번째인지 기록할 변수 count와 해당 줄의 길이를 기록할 변수 capacity를 사용한다. 한 개의 for 반복문 속에서 매 반복마다 count를 확인하여 capacity 이상이면 줄바꿈을 추가하고 capacity를 두 배로 늘린다.

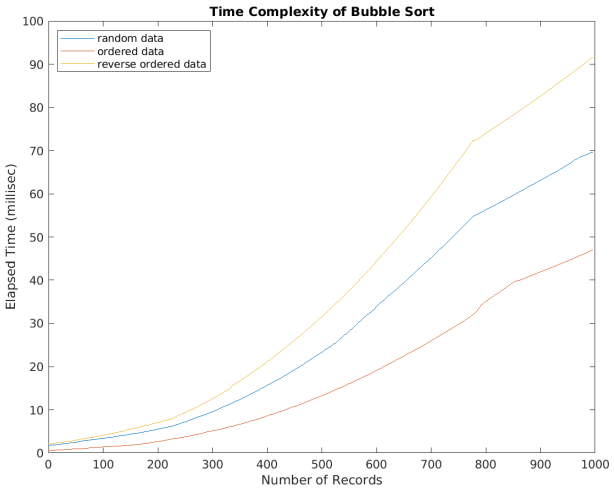
```
def printNum(n):
	capacity = 1
	count = 0

	for i in range(1, n + 1):
		print i,
		count += 1

		if count >= capacity:
			print ''
			count = 0
			capacity *= 2
```

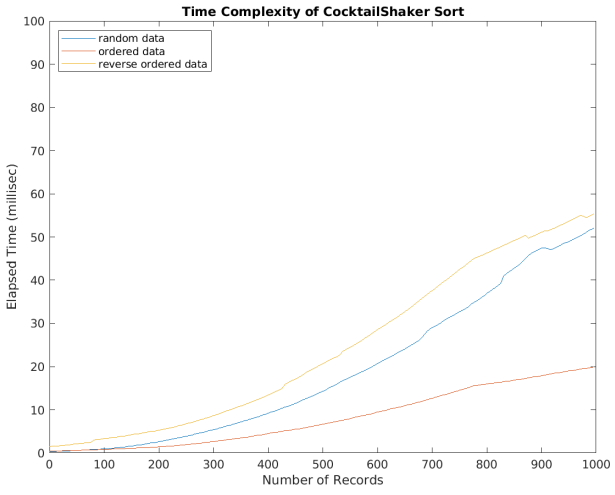
3.2 정렬 알고리즘 구현 및 실행시간 비교

- **cocktailShaker**

칵테일 정렬은 버블 정렬에 조금의 개선을 추가한 알고리즘이다. 버블정렬과 달리 매 회전마다 방향을 바꾼다.

버블 정렬과 성능을 비교하면 다음과 같다:

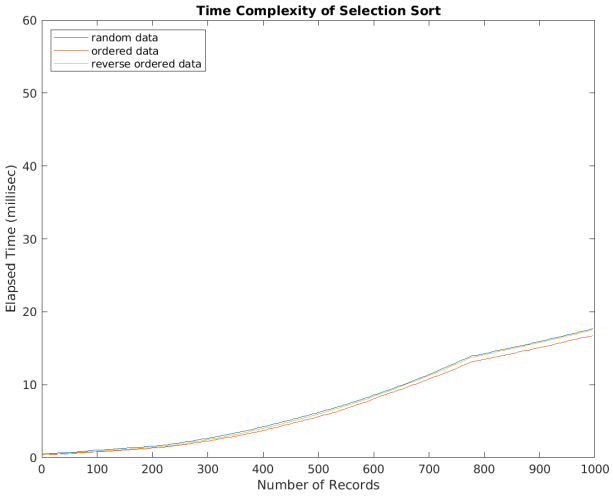


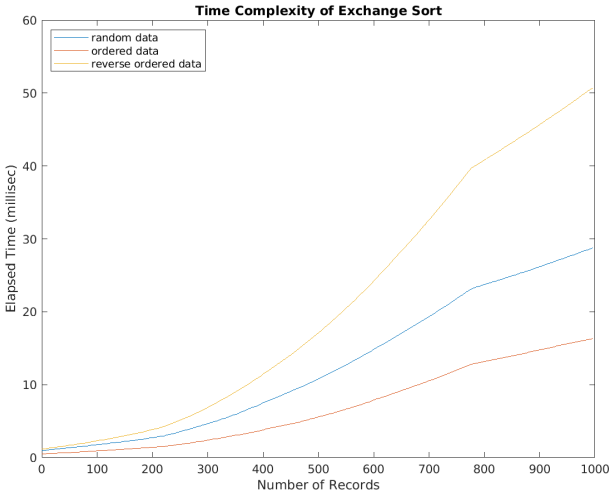


- **exchangeSort**

교환 정렬은 한 데이터에 대해 그 다음에 등장하는 데이터와 하나씩 비교하며 필요하다면 자리를 바꾸는 정렬이다.

선택 정렬과 실행 시간을 비교하면 다음과 같다:





4. 결론

칵테일 정렬은 방향을 바꾸는 것 만으로 큰 속도 향샹을 이루었다. 반면 교환 정렬은 구현이 매우 간단하지만 삽입 정렬과 선택 정렬에 비해 속도는 빠르지 않았다. 하지만 버블 정렬과 비교하면 교환 정렬이 조금 더 나은 성능을 보여준다.

5. 기타

전체 소스 코드는 GitHub 저장소(https://github.com/potados99/algorithm)에 업로드되어 있습니다.