SBS

23002 강민준

Bogo Sort란?

Bogo Sort란,

"정렬이 될 때까지 무작위로 섞는" 매우 비효율적인 정렬 알고리즘

1. 배열이 정렬되었는지 확인
2. 정렬되지 않았으면 셔플
3. 다시 확인

```
while not is_sorted(arr):
    shuffle(arr)
```

test01.py

test01.py

Bogo Sort

복잡도 종류	값
최악 시간복잡도	무제한 (랜덤화 버전), O((n+1)!) (결정화 버전)
최선 시간복잡도	O(n)
평균 시간복잡도	O((n+1)!)
공간복잡도	O(1)

기대 시도 수 n!

n이 조금만 커져도 정렬 불가능에 가깝다

n = 12일 때, 1초에 한번씩 정렬할 경우 15.18년 소요

SBS란? Smart Bogo Sort

SBS?

Bogo Sort를 일부 개선한 정렬

Bogo Sort를 부분적으로 활용하자

SBS?

1. 앞에서부터 차례대로 원소 확인

2. 현재의 원소가 뒤의 모든 원소보다 작거나 같으면 고정

3. 고정된 부분을 제외하고 무작위 셔플

```
while not is_sorted(arr):
    fixed = count_fixed_prefix(arr)
    arr[fixed:] = shuffle(arr[fixed:])
```

예시) [1, 4, 3, 2] 의 경우 1을 고정시키고 [4, 3, 2]에 대해서 재정렬

test02.py

test02.py

SBS

복잡도 종류	SmartBogoSort	BogoSort
최선 시간복잡도	O(n)	O(n)
평균 시간복잡도	O((n/2)!) ~ O(n^2)	O(n × n!)
최악 시간복잡도	O(n × n!)	무제한 (랜덤화 버전), O((n+1)!) (결정화 버전)
공간복잡도	O(1)	O(1)

실행 코드

main.cpp

다른 정렬과의 속도 비교

다른 정렬과의 속도 비교

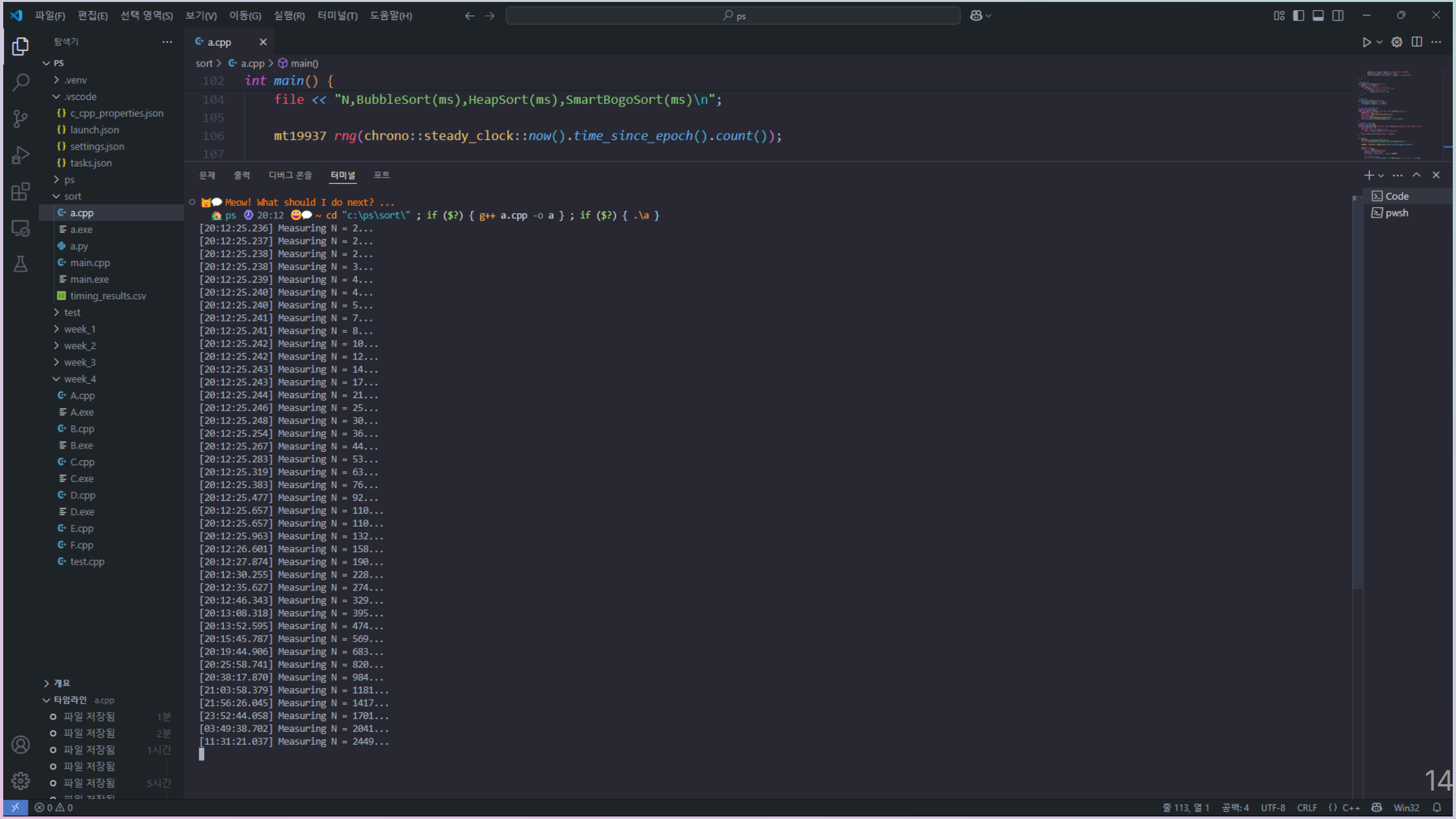
대조군 | 버블 정렬 | 힙 정렬

실행 코드

a.cpp

결과값

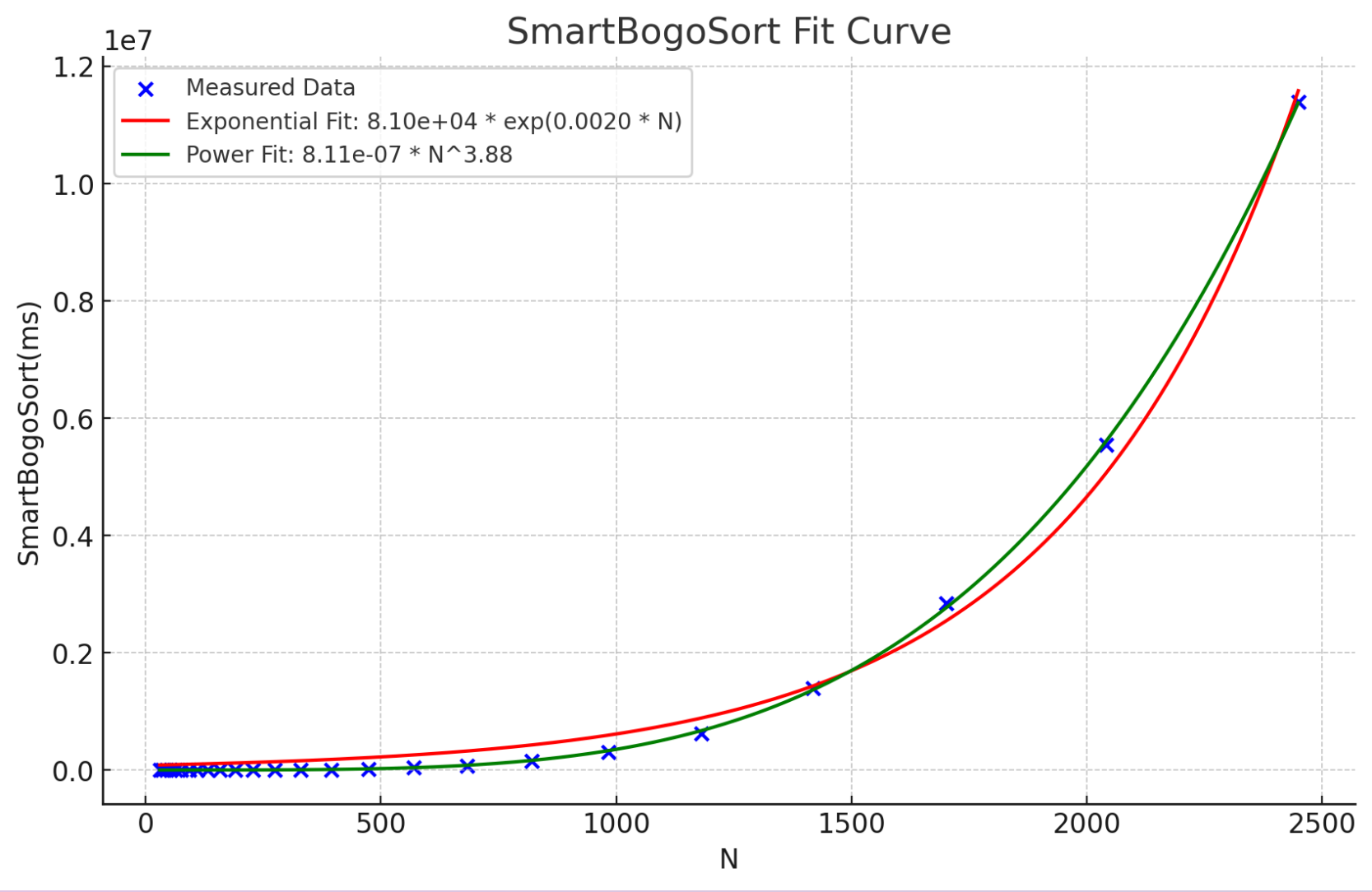
timing_results.csv


파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H)
ps
탐색기
a.cpp
sort > a.cpp > main()
102 int main() {
104 file << "N,BubbleSort(ms),HeapSort(ms),SmartBogoSort(ms)\n";
106 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
문제 출력 디버그 콘솔 터미널 포트
Meow! What should I do next? ...
ps 20:12 ~ cd "c:\ps\sort\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
[20:12:25.236] Measuring N = 2...
[20:12:25.237] Measuring N = 2...
[20:12:25.238] Measuring N = 2...
[20:12:25.238] Measuring N = 3...
[20:12:25.239] Measuring N = 4...
[20:12:25.240] Measuring N = 4...
[20:12:25.240] Measuring N = 5...
[20:12:25.241] Measuring N = 7...
[20:12:25.241] Measuring N = 8...
[20:12:25.242] Measuring N = 10...
[20:12:25.242] Measuring N = 12...
[20:12:25.243] Measuring N = 14...
[20:12:25.243] Measuring N = 17...
[20:12:25.244] Measuring N = 21...
[20:12:25.246] Measuring N = 25...
[20:12:25.248] Measuring N = 30...
[20:12:25.254] Measuring N = 36...
[20:12:25.267] Measuring N = 44...
[20:12:25.283] Measuring N = 53...
[20:12:25.319] Measuring N = 63...
[20:12:25.383] Measuring N = 76...
[20:12:25.477] Measuring N = 92...
[20:12:25.657] Measuring N = 110...
[20:12:25.657] Measuring N = 110...
[20:12:25.963] Measuring N = 132...
[20:12:26.601] Measuring N = 158...
[20:12:27.874] Measuring N = 190...
[20:12:30.255] Measuring N = 228...
[20:12:35.627] Measuring N = 274...
[20:12:46.343] Measuring N = 329...
[20:13:08.318] Measuring N = 395...
[20:13:52.595] Measuring N = 474...
[20:15:45.787] Measuring N = 569...
[20:19:44.906] Measuring N = 683...
[20:25:58.741] Measuring N = 820...
[20:38:17.870] Measuring N = 984...
[21:03:58.379] Measuring N = 1181...
[21:56:26.045] Measuring N = 1417...
[23:52:44.058] Measuring N = 1701...
[03:49:38.702] Measuring N = 2041...
[11:31:21.037] Measuring N = 2449...
Code
pwsh
줄 113, 열 1 공백: 4 UTF-8 CRLF C++ Win32

다른 정렬과의 속도 비교

알고리즘	최선	평균	최악	공간 복잡도
SBS	O(n)	O((n/2)!) ~ O(n^2)	O(n × n!)	O(1)
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Heap Sort	O(n log n)	O(n log n)	O(n log n)	O(1)

SBS의 추세선



지수 함수 피팅

$$y = 8.10 \times 10^{4} \cdot e^{0.0020 \cdot N}$$

거듭제곱 함수 피팅

$$y = 8.11 \times 10^{-7} \cdot N^{3.88}$$

SBS 추세선

🔴 지수 함수 피팅 (Exponential Fit)

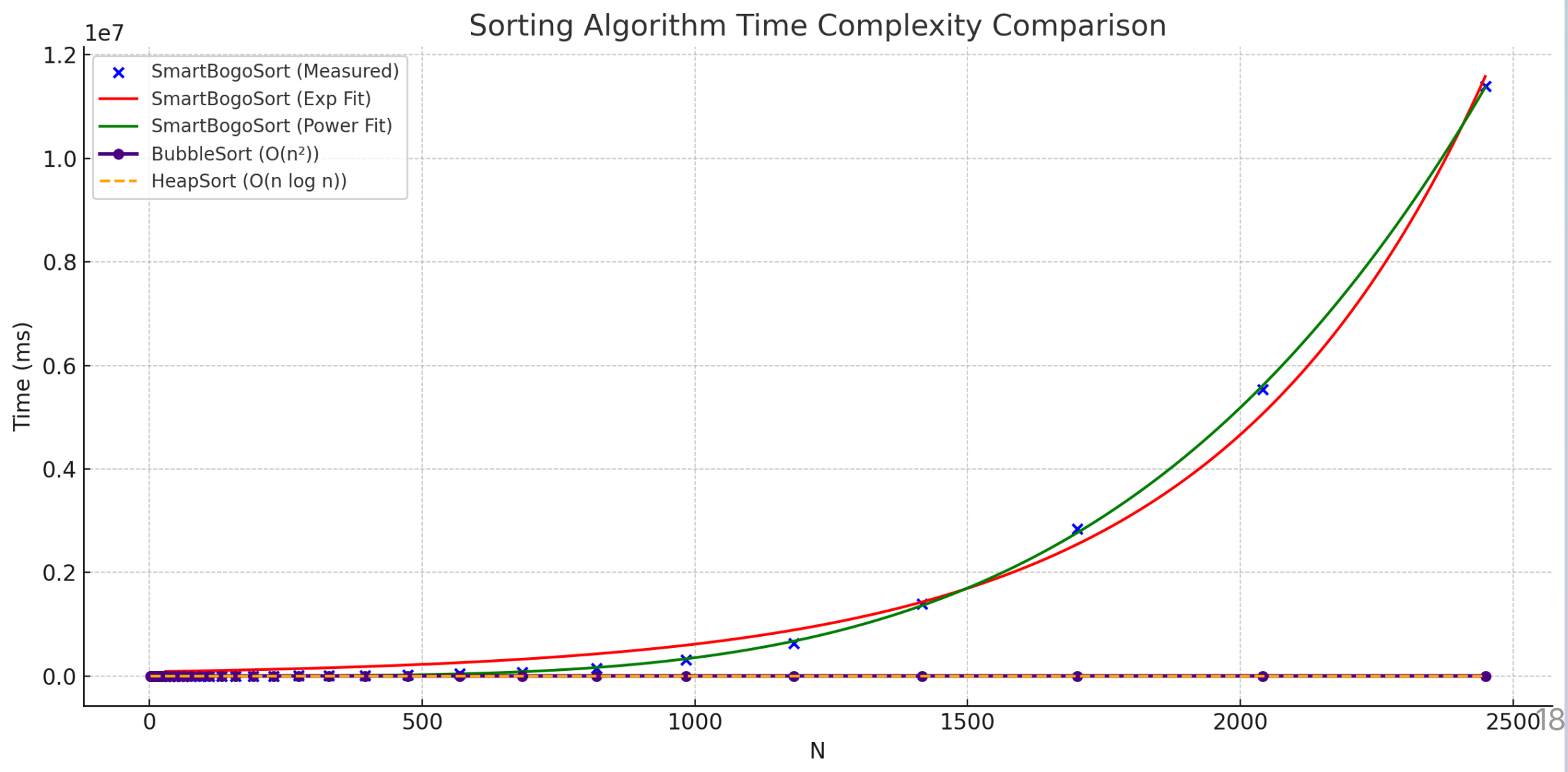
$$y = 4.09 \times 10^{-3} \cdot e^{0.0121 \cdot N}$$

🟢 거듭제곱 함수 피팅 (Power Fit)

$$y = 2.11 \times 10^{-10} \cdot N^{5.98}$$

거듭제곱 함수 피팅의 R^2 값이 더 높아, 실제 평균 실행 시간에 더 잘 맞음

다른 정렬과의 비교


Sorting Algorithm Time Complexity Comparison
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Time (ms)
SmartBogoSort (Measured)
SmartBogoSort (Exp Fit)
SmartBogoSort (Power Fit)
BubbleSort (O(n²))
HeapSort (O(n log n))
0
500
1000
1500
2000
2500
N

매우 비효율적임을 확인.

시간복잡도 계산

기대 시간복잡도

SBS는 길이 n짜리 배열에 대해 다음과 같은 수식을 따름.

$$T(n) = \sum_{k=1}^{n}(n-k)! = (n-1)! + (n-2)! + \cdots + 1! + 0!$$

평균적 근사

평균적으로 절반 정도가 고정된다고 가정하면

$$T(n) \approx O\left(\left(\frac{n}{2}\right)!\right)$$

최악의 경우

$$T(n) = O(n \cdot n!)$$

예시: (n = 10)일 때 계산

- T(10) = $\sum_{k=1}^{10}(10-k)! = 9! + 8! + \cdots + 0! = \mathbf{409{,}114}$
- 평균 고정 가정: $\left(\frac{10}{2}\right)! = 5! = \mathbf{120}$
- 최악의 경우: $10 \cdot 10! = \mathbf{36{,}288{,}000}$

해석 요약

구분	수식	계산 예시 (n=10)
기대값	$\sum_{k=1}^{n}(n-k)!$	$409,114$
평균 근사	$O\left(\left(\frac{n}{2}\right)!\right)$	120
최악의 경우	$O(n \cdot n!)$	$36,288,000$

감사합니다