Assignment 2

Sub-set Sum

Matjaž Mav (63130148) May 3, 2020

1 DYN

This approach has time complexity of $\mathcal{O}(nk)$ and space complexity of $\mathcal{O}(k)$. It is greatly effected by the size of array ***A*** and/or targeted sum ***k***. If we introduce big numbers this problem becomes really hard for this approach. Example of such problem would be:

```
5
10000000
2000000
2000000
2000000
2000000
2000000
```

2 EXH

This approach has time and space complexity of $\mathcal{O}(2^n)$. The worst case scenario is when the input is a long array ***A*** of some relatively small numbers and some really big targeted sum ***k***. If the targeted sum ***k*** is greater or equal then sum of the whole array ***A***, this approach will perform exhaustive search without any kind of pruning. Example of such problem would be:

```
3000000
6000000
1
2
3
1
2
3
...
1
2
3
```

3 GREEDY

This approach has time complexity of $\mathcal{O}(n \log n)$ and space complexity of $\mathcal{O}(n)$. In the worst case this algorithm returns at most 2 times worse result. Example of such problem would be:

```
3
1000000
500001
500000
500000
```

In this case optimal result is 1000000 but the algorithm would return 500001. The result is almost 2 times worse then the optimal ($500001 \approx 1000000/2$).

4 FPTAS

This is ϵ-approximation. Its execution time is greatly effected by the parameter ϵ. We have compared how the ϵ parameter effect the execution time of this algorithm. We have generated random inputs using code listed in the Listing 1. We observed (Figure 1) that with smaller values of ϵ this algorithm performs slower and has higher standard deviation.

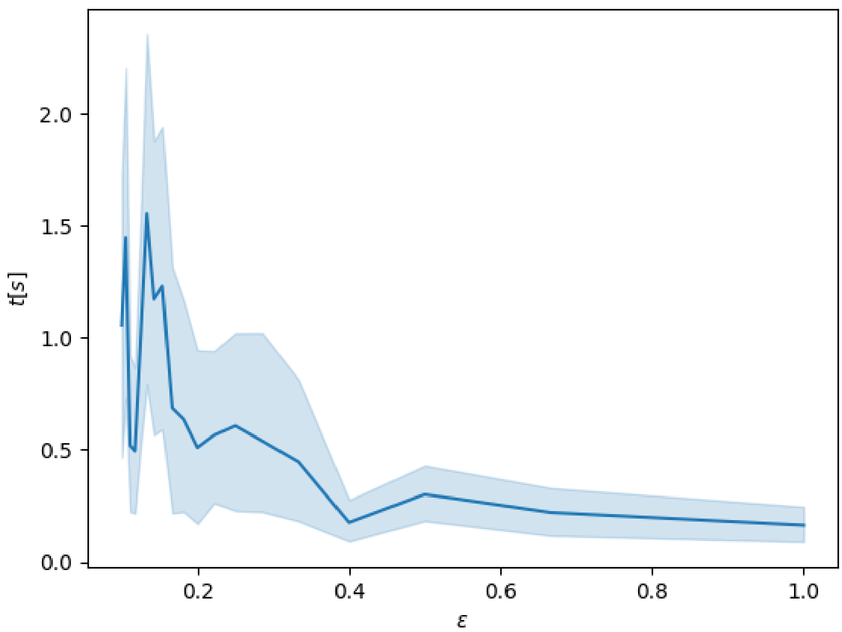

Figure 1: Execution time compared to the parameter ϵ

Listing 1: Random problem generator

```
def generate(n: int) -> Tuple[List[int], int]:
  A = []
  for _ in range(n):
    A.append(randint(1, int(n/2)))

  k = sum(sample(A, int(n*random())))
  return (A, k)
```