School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4032A
Applications of Algorithms
Assignment 5

Moses Katlego Modupi (1614669)

October 21, 2020

1 Part A

1.1 Test Input generation

We ran Graham Scan and Jarvis march on n randomly generated floating point data points. We ran multiple experiments where we varied n from 30 000 to 1 200 000 and increasing n by 30 000. For each n, we ran the 2 algorithms 30 times and recorded the average runtime. The graphs below are what resulted from these tests. Figure 1a shows the runtime for Graham scan and it has the expected runtime of $O(nlogn)$. Figure 1b shows the runtime for Jarvis march. We expect the runtime for Jarvis march to be $O(nh)$ (h being the number of vertices in the convex hull). The red plot hows runtime/h, which is linear as expected. Furthermore, the purple plot shows that although, Jarvis march have a worst case complexity of $O(n^2)$, It is very unlikely to have a case where h is close to n. This is shown by how the purple curve closely resembles the best case of $O(n)$. Figures 1c,d show how in effective Jarvis March is in practice compared to Graham Scan.

1.2 Graphs

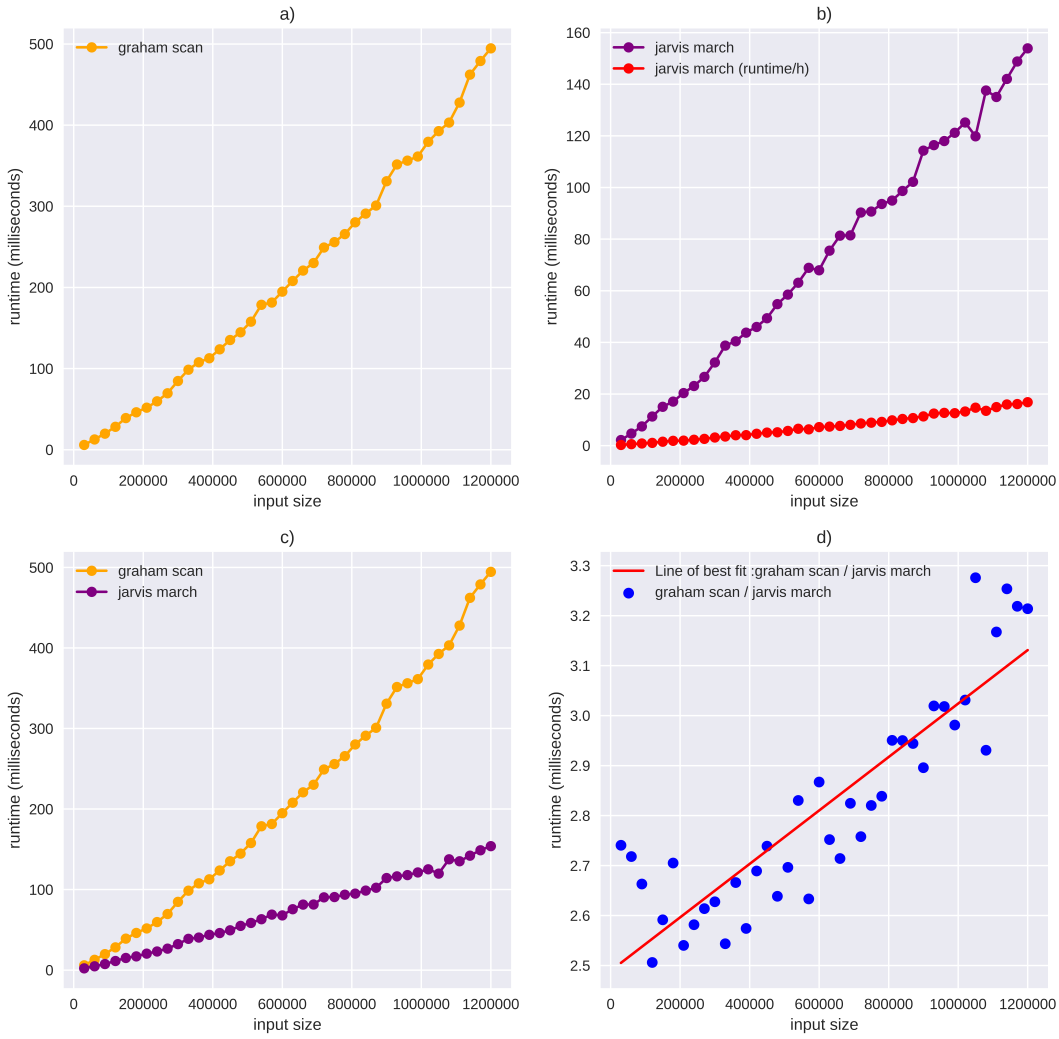

Figure 1: a)-Graham Scan Runtime | b)-Jarvis March Runtime | c)-Graham Scan vs Jarvis March | d) Jarvis March speedup over Graham Scan.

2 Part B

Since the points in the polygon are already listed in a counterclockwise order, we can use Graham scan without the sorting part which would result in a complexity of $O(n)$.

Algorithm 1: CH_star_polygon(P, n)

```
1:  S = empty_stack()
2:  S.push(P[0])
3:  S.push(P[1])
4:  S.push(P[2])
5:  for i = 3 to n do
6:      while the angle formed by points S.NextToTop(), S.Top() ,and P[i] makes a non-left turn do
7:          S.pop()
8:      end while
9:      S.push(P[i])
10: end for
11: return S;
```