WITS UNIVERSITY

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

COMS4032A
Applications of Algorithms
Assignment 5

Moses Katlego Modupi (1614669)

October 27, 2020

1 Part A

1.1 Test Input generation

We ran Graham Scan and Jarvis march on n randomly generated floating point data points using the java Random(). We ran multiple experiments where we varied n from 40 000 to 1 600 000 and increasing n by 40 000. For each n, we ran the 2 algorithms 20 times and recorded the average runtime. The graphs below are what resulted from these tests. Figure 1 shows the runtime for Graham scan and we see runtimes that resembles $O(nlogn)$. Figure 2a shows the runtime for Jarvis march as well as a curve representing $\frac{jarvis\_runtime}{h}$ (h being the number of vertices in the convex hull). We expect the runtime for Jarvis march to be $O(nh)$. Figure 2b shows an almost $log(n)$ relation between n and h and this is consistent to the graphs we obtained in 2a.

1.2 Graphs

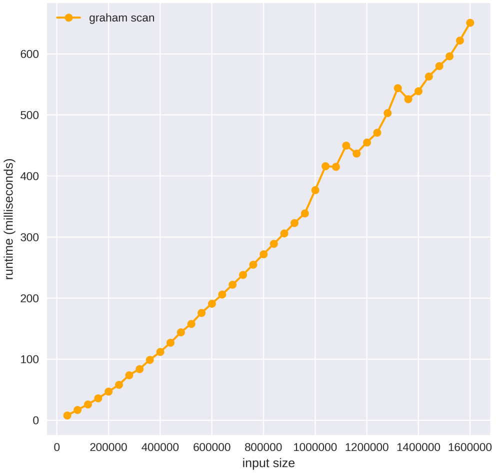

Figure 1: a) Graham Scan Runtime

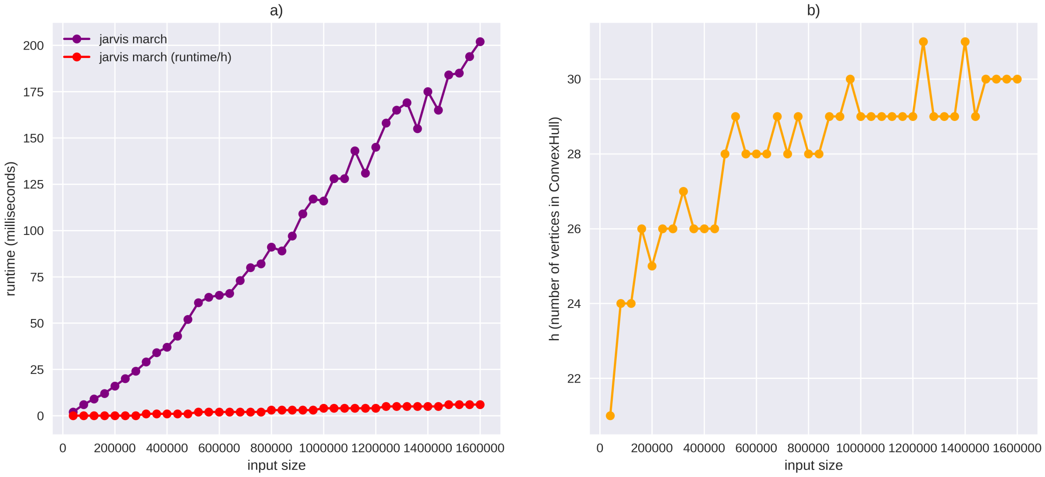

Figure 2: a) Jarvis March Runtime b) n vs h

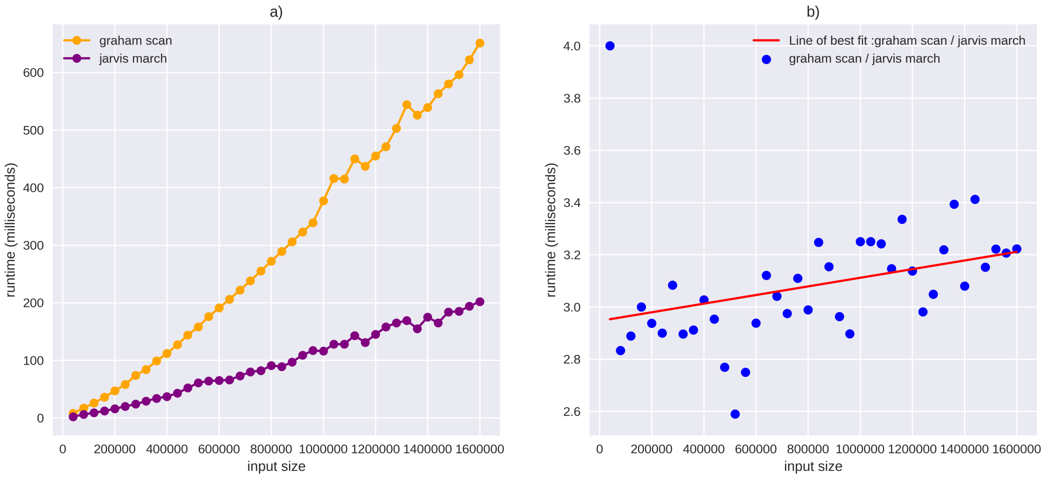

Figure 3: a) Graham Scan vs Jarvis March b) Jarvis March speedup over Graham Scan

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)

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