# Yapay Zeka

# A*
# and
# Best First Search

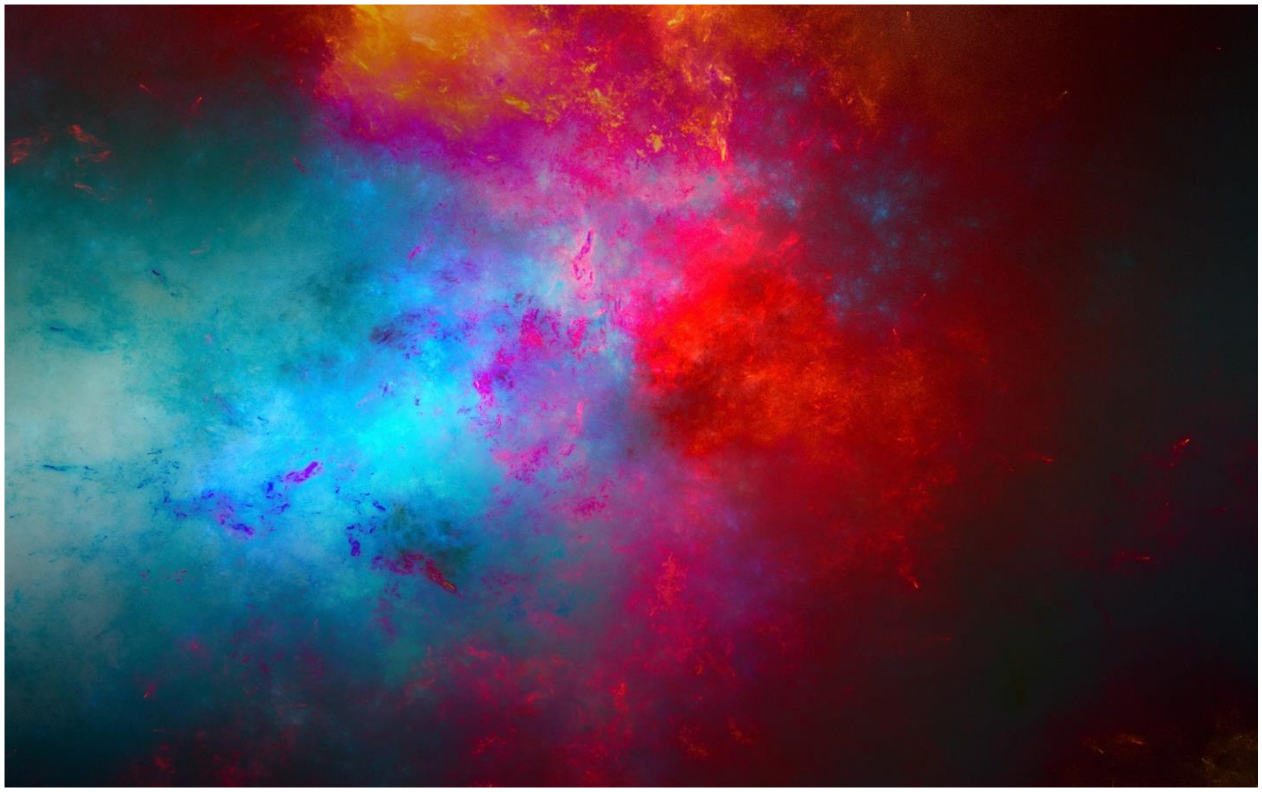
Mustafa Katipoğlu

16011084

# Images and Outputs

(Some Images Zoomed In)

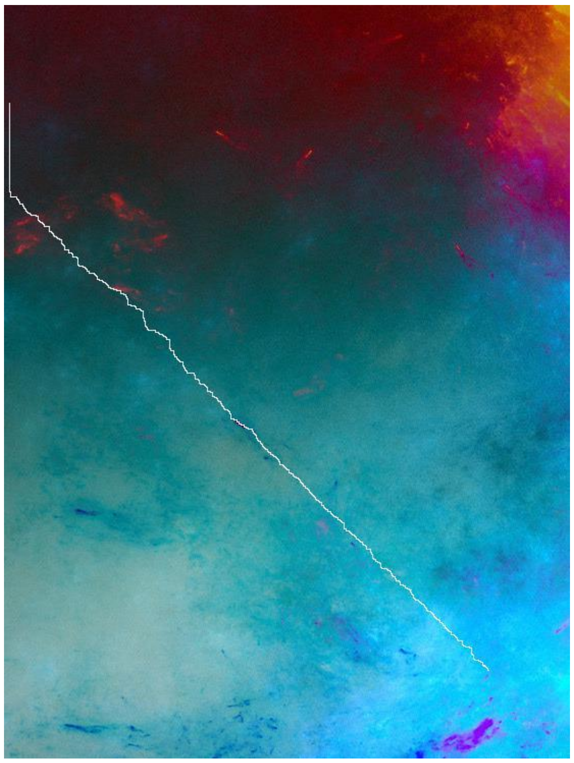
# Original Input:



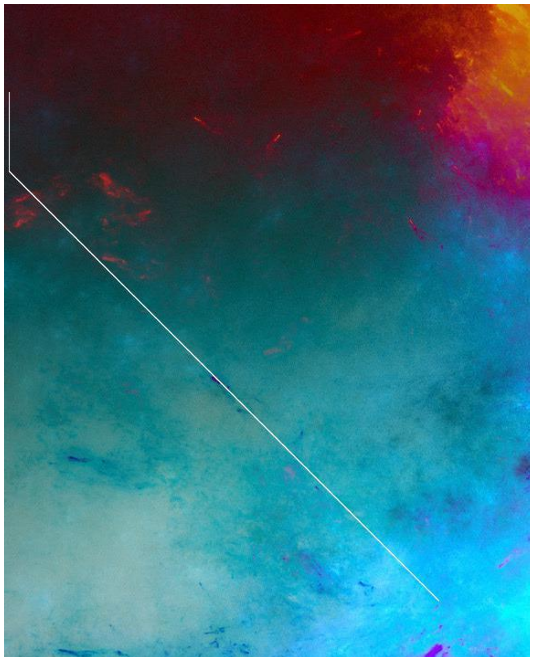
## User Input

```
Initial Configurations: Start:(5,96) ::: Target:(475,653)
Start(5,96) Target(475,653)
Max Size Seen: BFS:3082, BFS_2:3082, A*:3088, A*_2:3088
Number of Deletions: BFS:1028, BFS_2:1028, A*:1030, A*_2:1030
Runtime(in seconds):
        BFS:0.20, BFS_2=20.84, A*=0.21, A*_2=23.00
```

A*



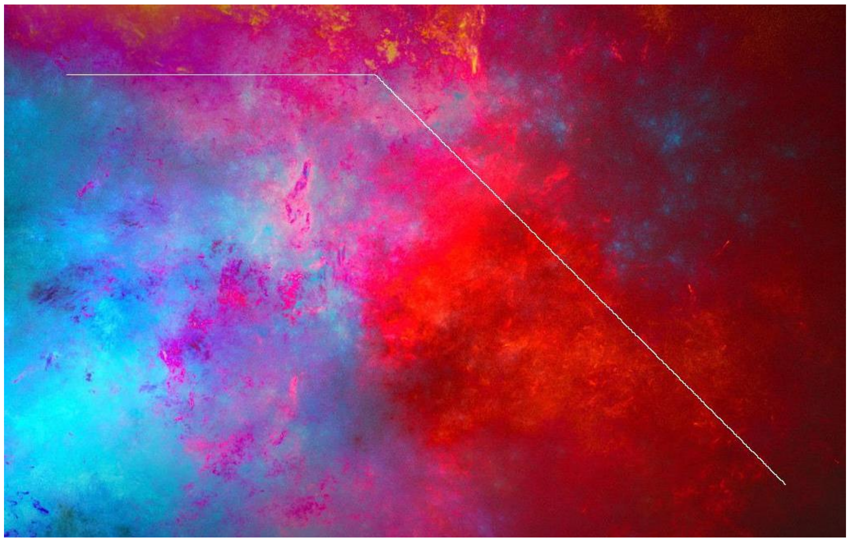
BFS

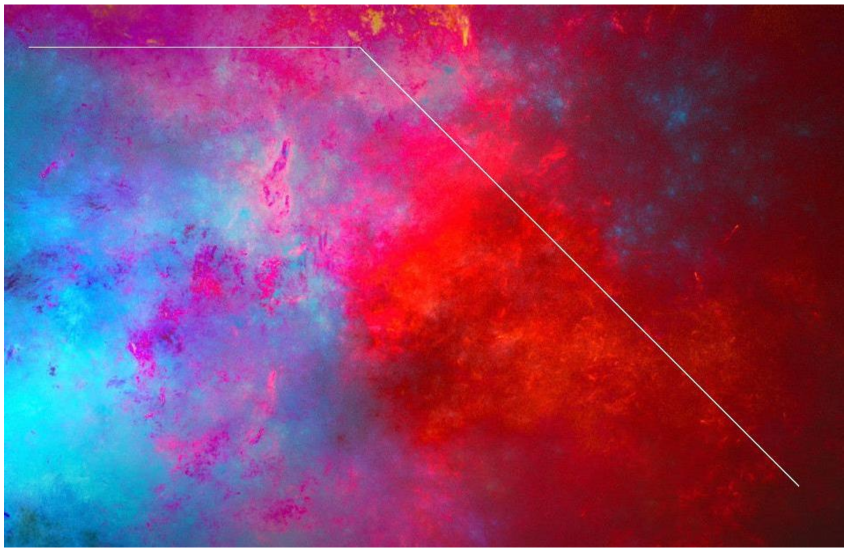


Max Queue Size and Number of Deletions in between algorithms almost same.

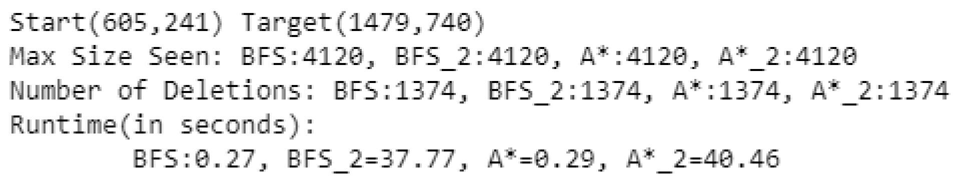
# Random Points

A* First, BFS Second



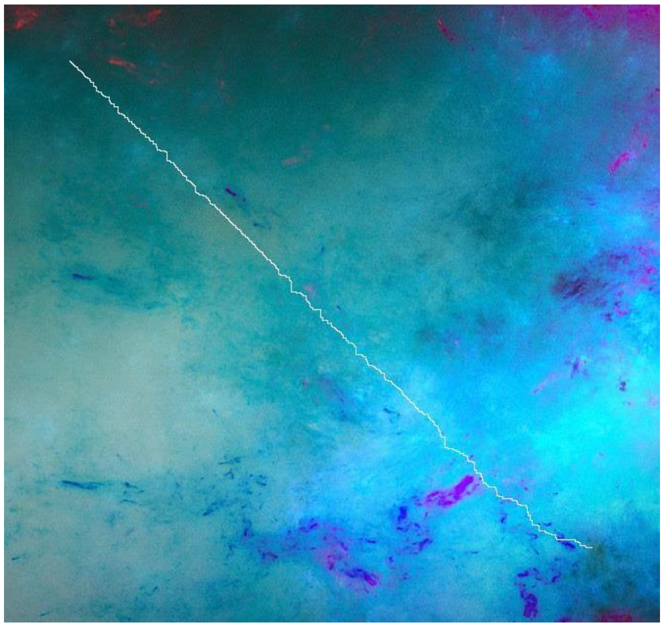


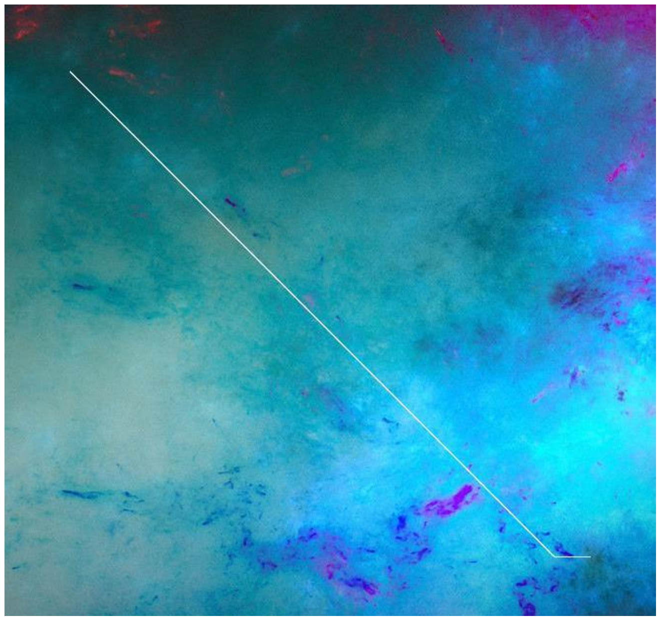
A* First, BFS Second



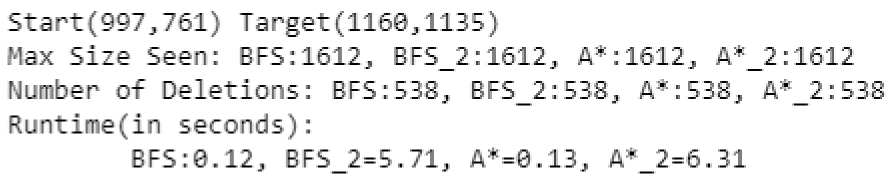
```
Start(605,241) Target(1479,740)
Max Size Seen: BFS:4120, BFS_2:4120, A*:4120, A*_2:4120
Number of Deletions: BFS:1374, BFS_2:1374, A*:1374, A*_2:1374
Runtime(in seconds):
        BFS:0.27, BFS_2=37.77, A*=0.29, A*_2=40.46
```

Almost same path but BFS straight, A* is notched.



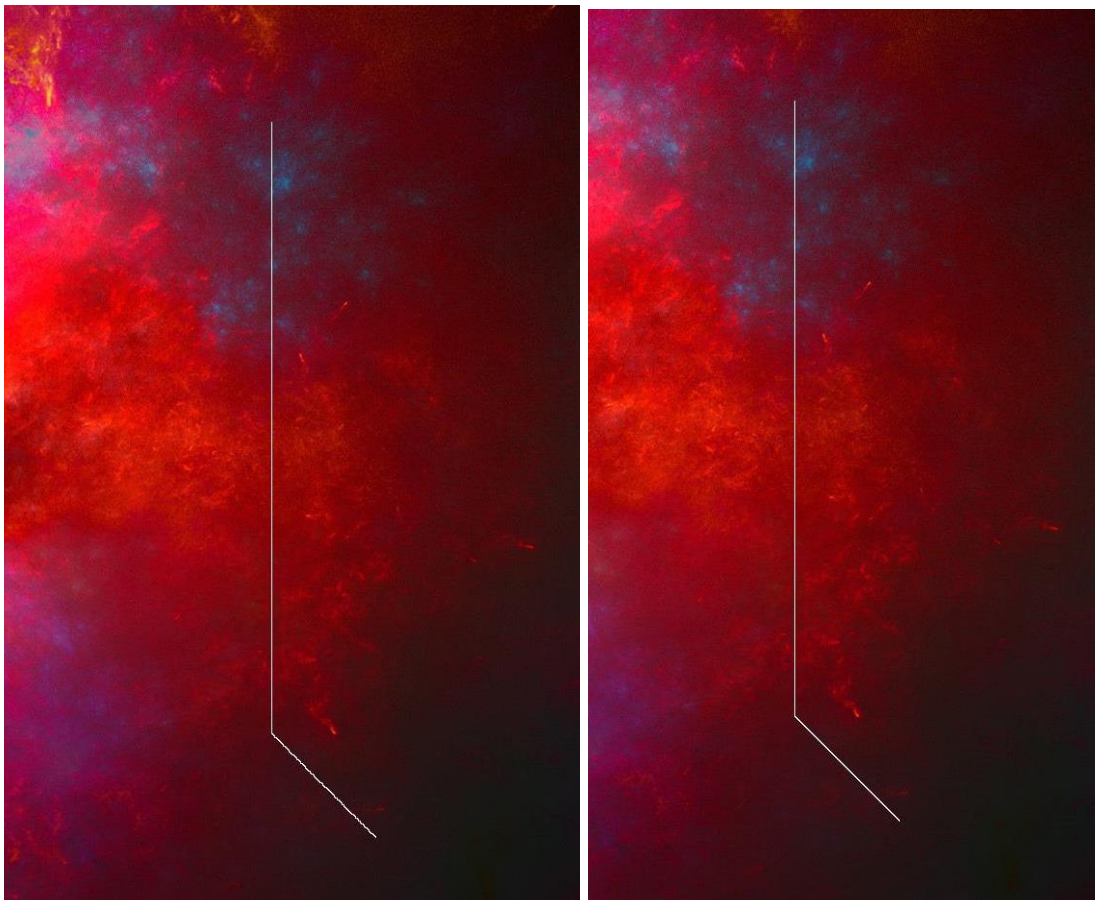


A* First, BFS Second

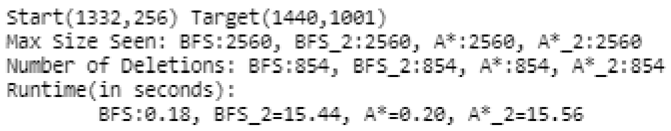


```
Start(997,761) Target(1160,1135)
Max Size Seen: BFS:1612, BFS_2:1612, A*:1612, A*_2:1612
Number of Deletions: BFS:538, BFS_2:538, A*:538, A*_2:538
Runtime(in seconds):
        BFS:0.12, BFS_2=5.71, A*=0.13, A*_2=6.31
```

Again almost same path but BFS straight, A* is notched. But also both of them have same number of max-queue-size and number of deletions on all algorithms.

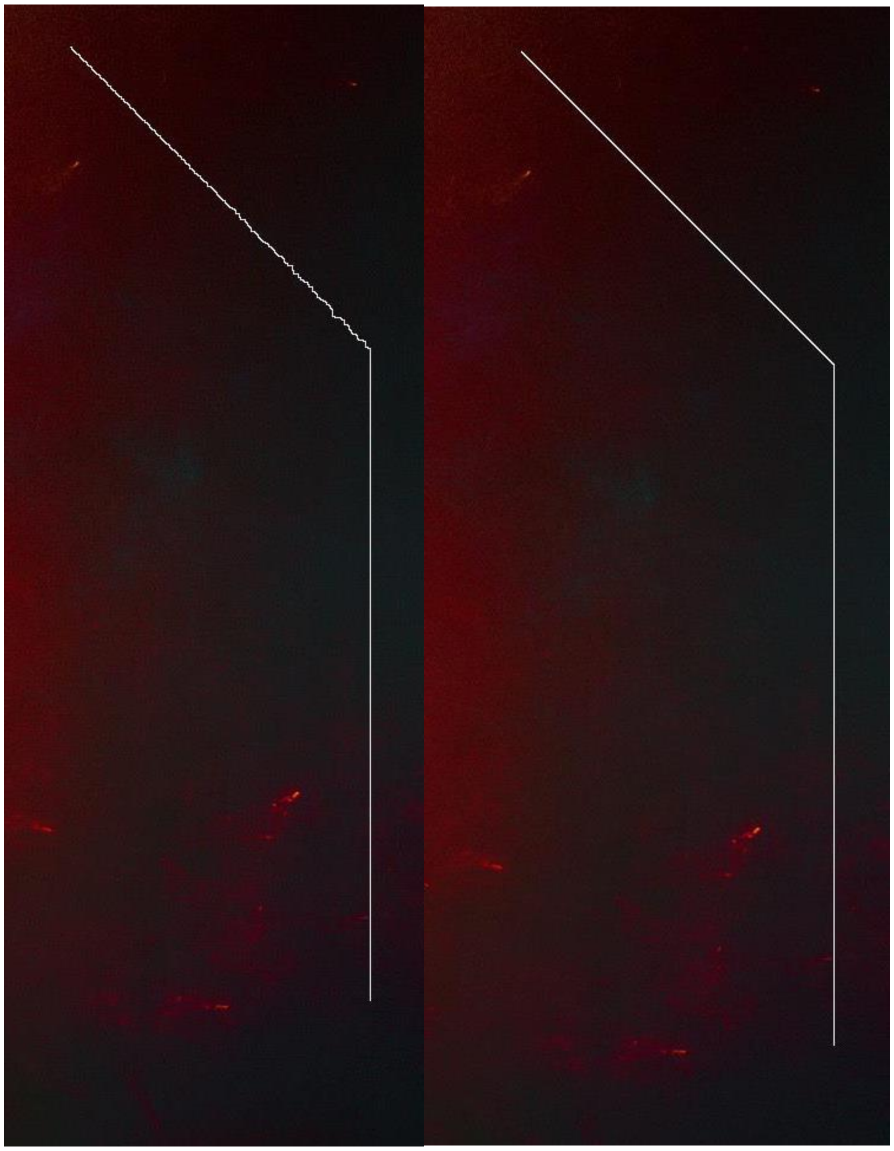


A* Left, BFS Right

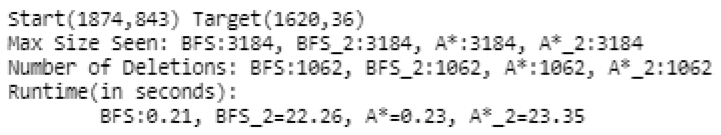


```
Start(1332,256) Target(1440,1001)
Max Size Seen: BFS:2560, BFS_2:2560, A*:2560, A*_2:2560
Number of Deletions: BFS:854, BFS_2:854, A*:854, A*_2:854
Runtime(in seconds):
        BFS:0.18, BFS_2=15.44, A*=0.20, A*_2=15.56
```

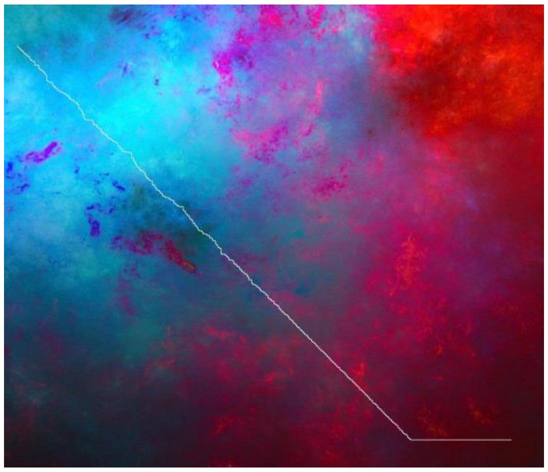
Again almost same path but BFS straight, Last part of A* is notched. Same Max heap-size and number of deletions.

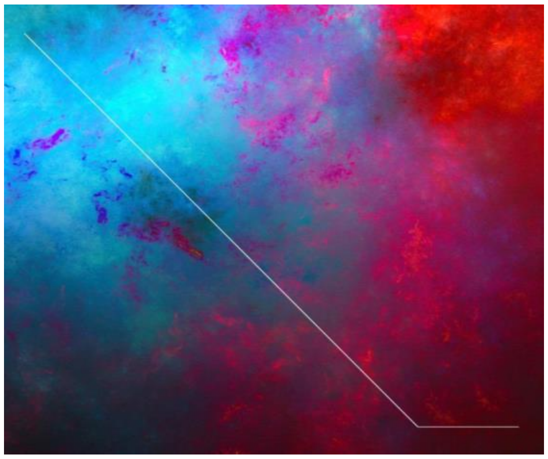


A* Left, BFS Right

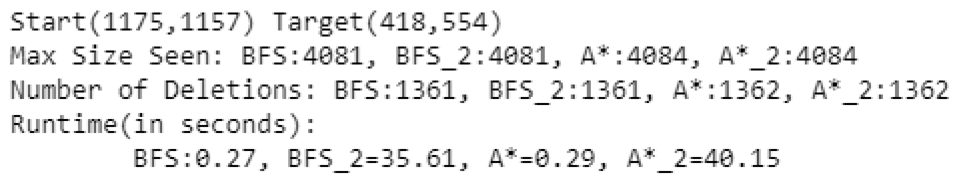


```
Start(1874,843) Target(1620,36)
Max Size Seen: BFS:3184, BFS_2:3184, A*:3184, A*_2:3184
Number of Deletions: BFS:1062, BFS_2:1062, A*:1062, A*_2:1062
Runtime(in seconds):
        BFS:0.21, BFS_2=22.26, A*=0.23, A*_2=23.35
```



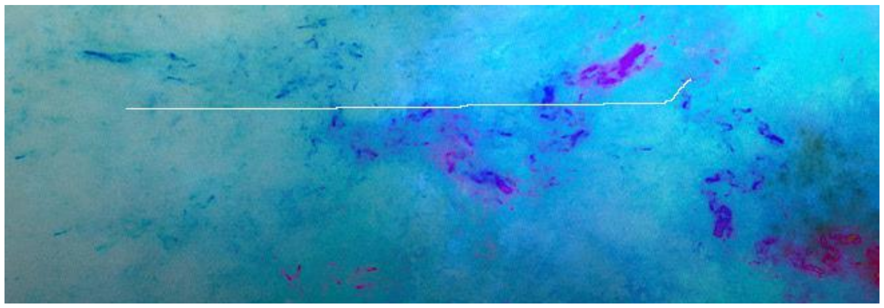


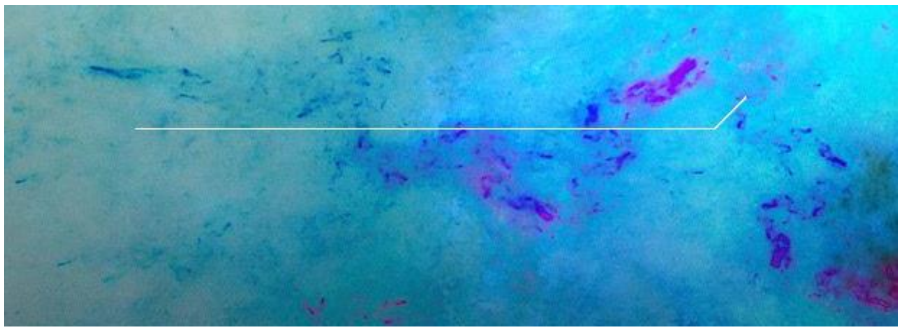
A* First, BFS Second



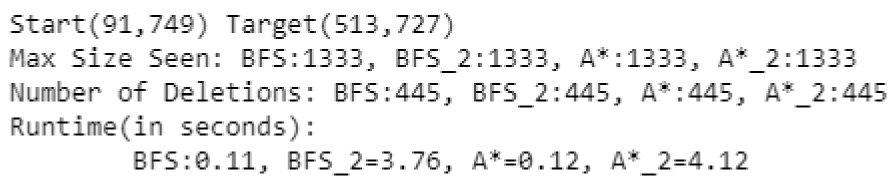
```
Start(1175,1157) Target(418,554)
Max Size Seen: BFS:4081, BFS_2:4081, A*:4084, A*_2:4084
Number of Deletions: BFS:1361, BFS_2:1361, A*:1362, A*_2:1362
Runtime(in seconds):
        BFS:0.27, BFS_2=35.61, A*=0.29, A*_2=40.15
```

Again almost same path but BFS straight, A* is notched. Max-queue-size and number of deletions on BFS and A* is almost same, just a few difference.



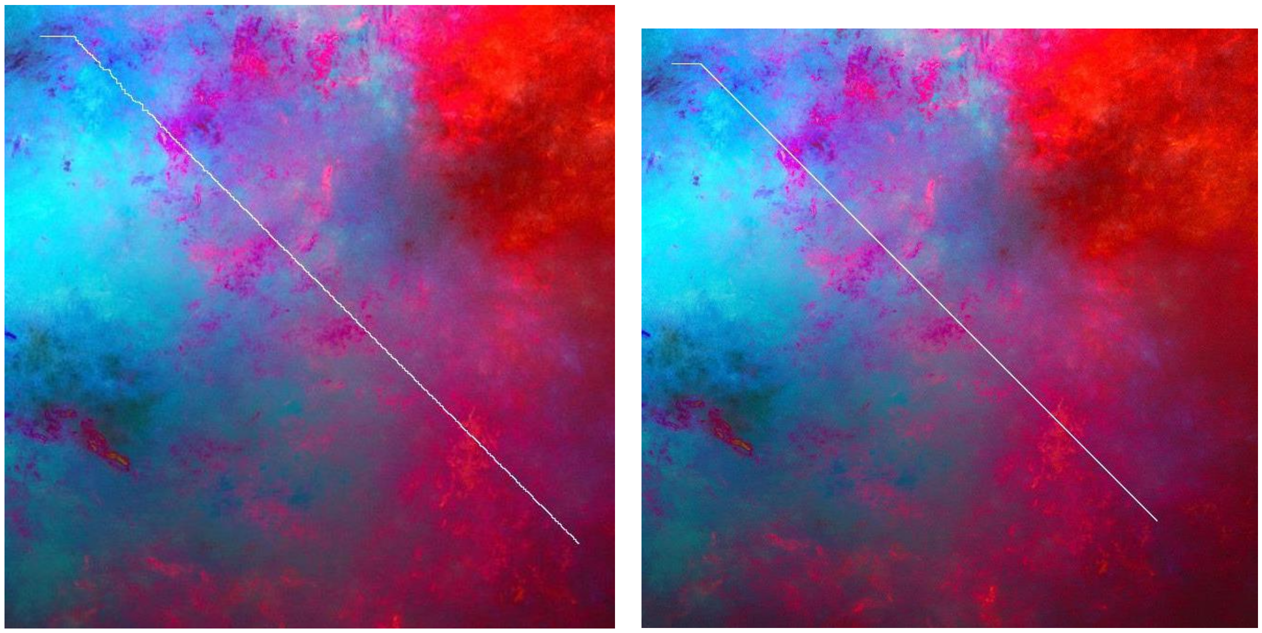


A* First, BFS Second

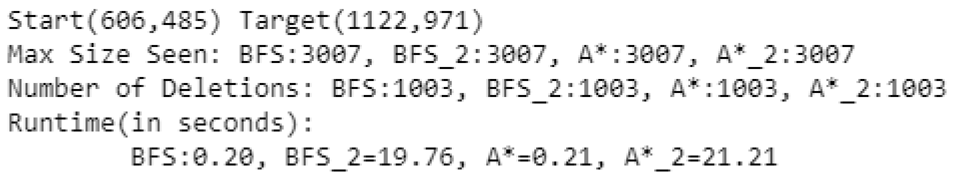


```
Start(91,749) Target(513,727)
Max Size Seen: BFS:1333, BFS_2:1333, A*:1333, A*_2:1333
Number of Deletions: BFS:445, BFS_2:445, A*:445, A*_2:445
Runtime(in seconds):
        BFS:0.11, BFS_2=3.76, A*=0.12, A*_2=4.12
```

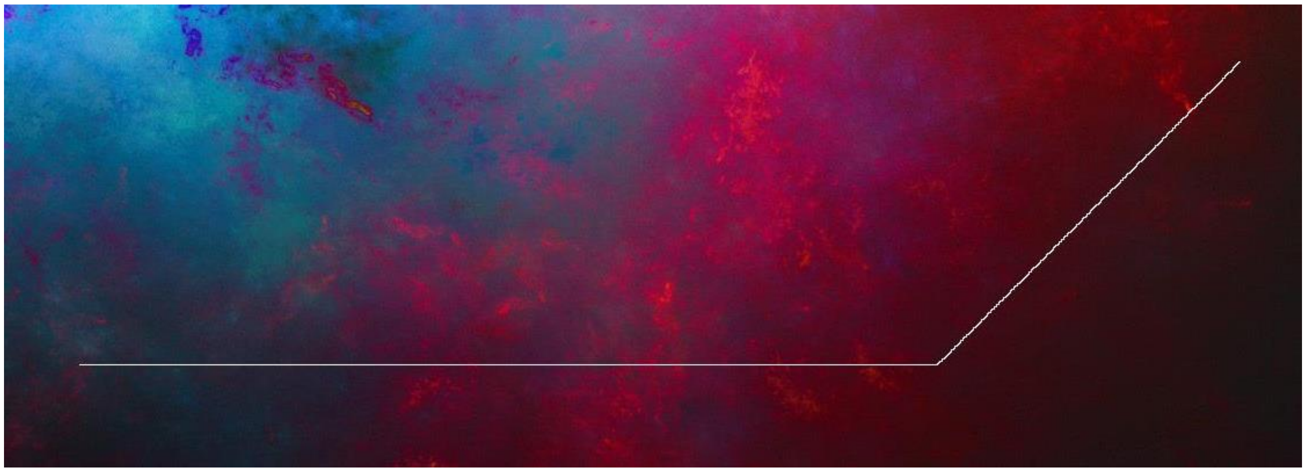
Same path in little difference in the way. Max-queue-size and number of deletions on BFS and A* is same.

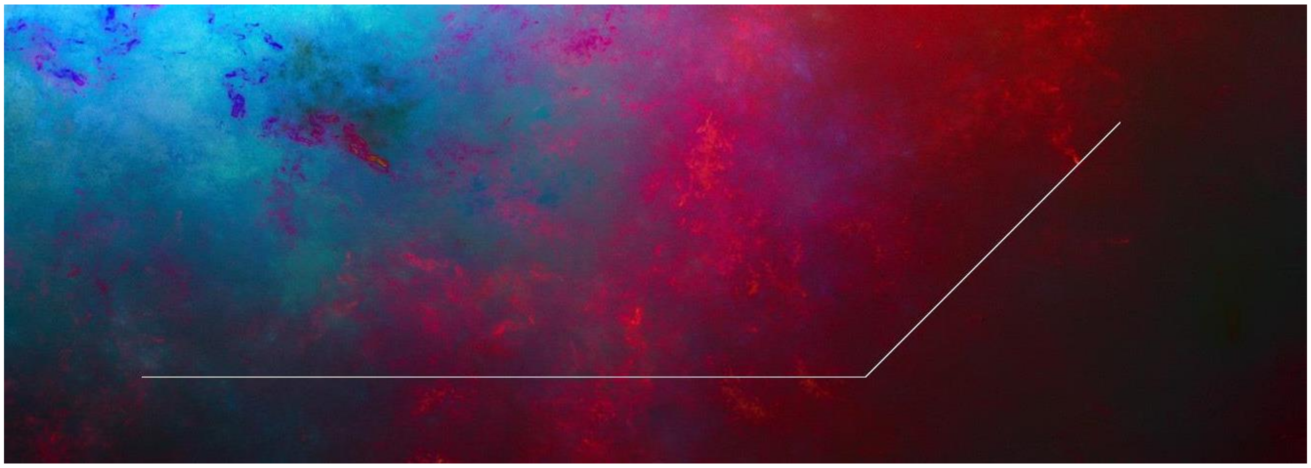


A* Left, BFS Right

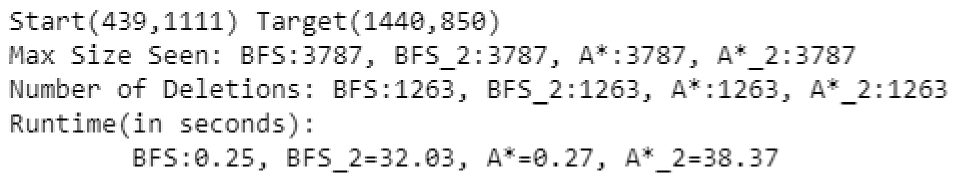


```
Start(606,485) Target(1122,971)
Max Size Seen: BFS:3007, BFS_2:3007, A*:3007, A*_2:3007
Number of Deletions: BFS:1003, BFS_2:1003, A*:1003, A*_2:1003
Runtime(in seconds):
        BFS:0.20, BFS_2=19.76, A*=0.21, A*_2=21.21
```



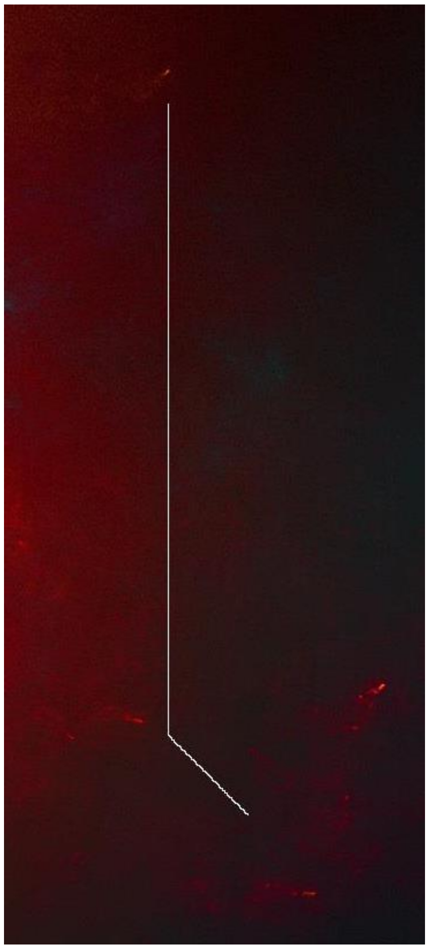


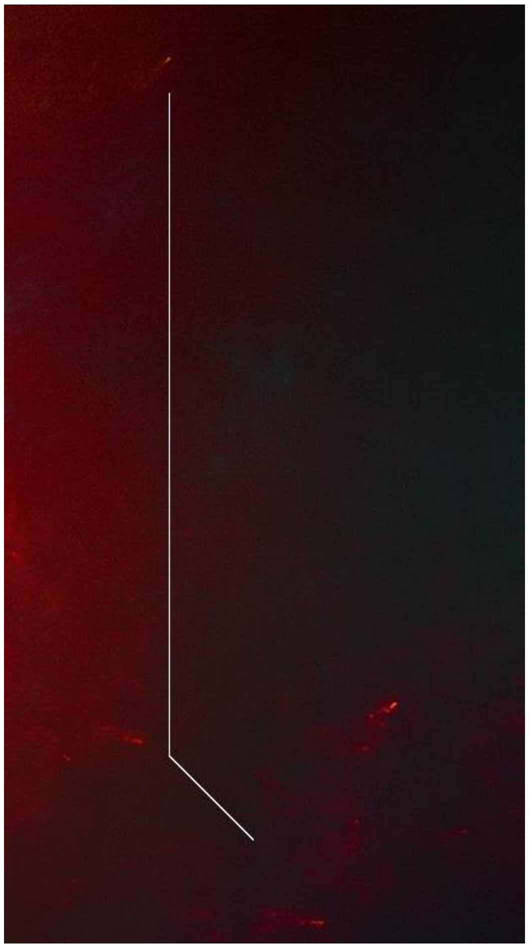
A* First, BFS Second

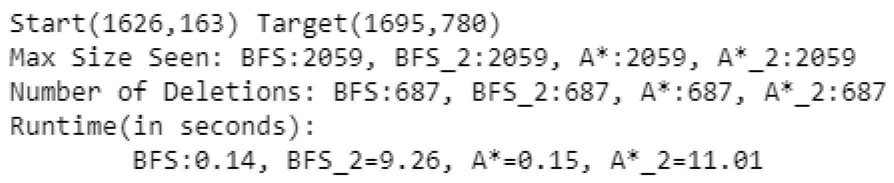


```
Start(439,1111) Target(1440,850)
Max Size Seen: BFS:3787, BFS_2:3787, A*:3787, A*_2:3787
Number of Deletions: BFS:1263, BFS_2:1263, A*:1263, A*_2:1263
Runtime(in seconds):
        BFS:0.25, BFS_2=32.03, A*=0.27, A*_2=38.37
```

As image gets darker, notches on the A* path goes lower and the path is more and more looks like BFS path.





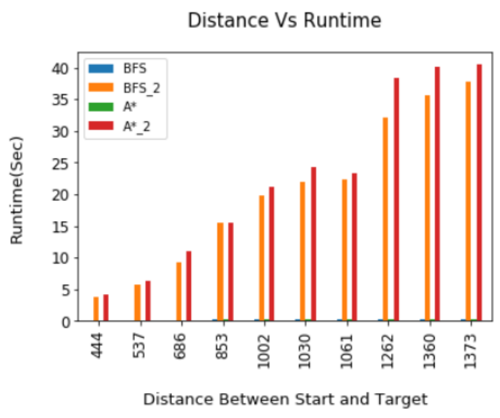


```
Start(1626,163) Target(1695,780)
Max Size Seen: BFS:2059, BFS_2:2059, A*:2059, A*_2:2059
Number of Deletions: BFS:687, BFS_2:687, A*:687, A*_2:687
Runtime(in seconds):
        BFS:0.14, BFS_2=9.26, A*=0.15, A*_2=11.01
```

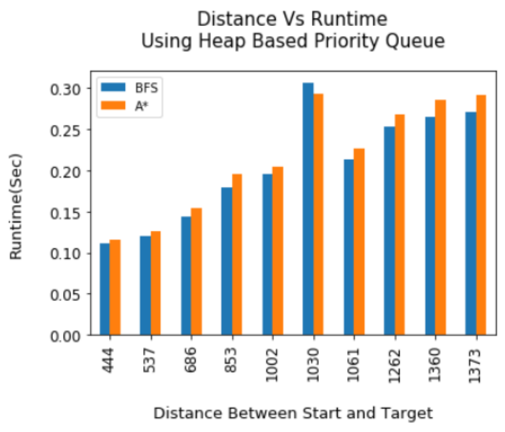
This one supports last statement too, as image gets darker, notches on the A* path goes lower and the path is more and more looks like BFS path.

## Comparing Algorithms

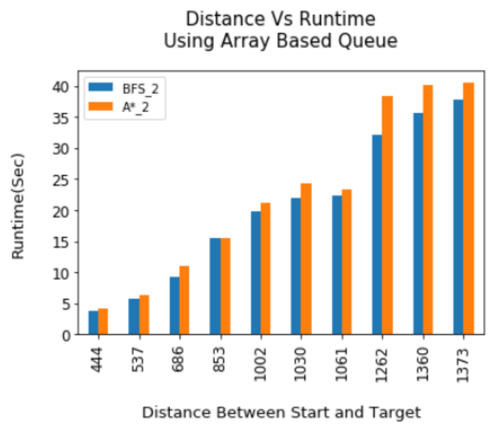
## Case 1: When Distances are Random



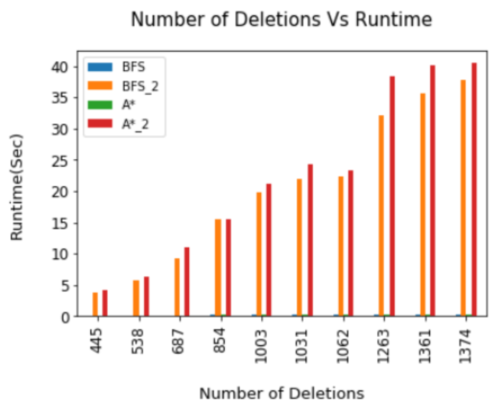
**PLOT#1**:Distance Vs Runtime
**INSIGHT#1**: As distance between start and target increase,the runtime of the algorithms that have been implemented using array based queue(BFS_2, A*_2) outweigh the runtime of the algorithms that have been implemented using heap based priority queue(BFS,A*)
**INSIGHT#1.1**: The runtime of the algorithms that have been implemented using array based queue(BFS_2, A*_2) is so inefficient in compared to the the algorithms that have been implemented using heap based priority queue(BFS,A*)



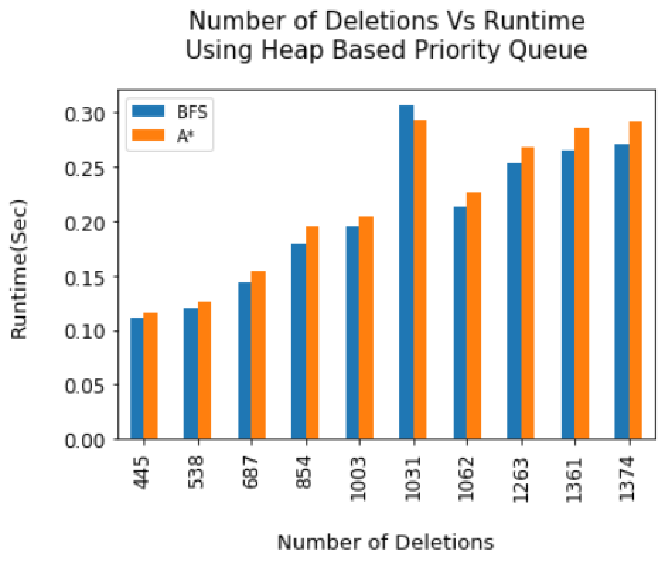
**PLOT#2**:Distance Vs Runtime Using Heap Based Priority Queue
**INSIGHT#2**: When heap based priority queue used, as distance between start and target increase, the runtime increases linearly
For Example when distance goes frm 444 to 1373 (approximately 3 fold), runtime also increases 3 fold
**INSIGHT#2.1**: When heap based priority queue used, The Algorithm chosen does not differentiates the runtime as much as the distance change differentiates



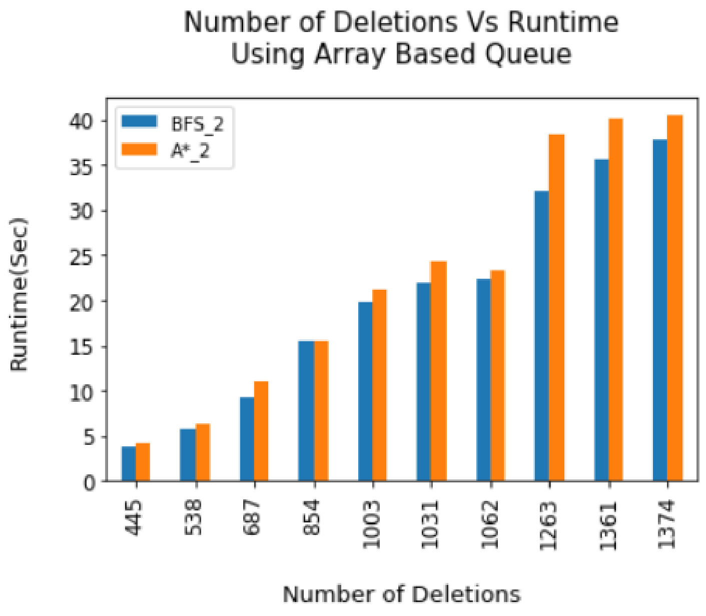
**PLOT#3**:Distance Vs Runtime Using Array Based Queue
**INSIGHT#3**: As distance between start and target increase, the runtime increases 2 times of the increase of the distance
For Example when distance goes from 444 to 1262 (approximately 4 fold), the runtime has increased approximately 7 fold, when distance goes from 686 to 1262 (approximately 2 fold), the runtime has increased approximately 3.5 fold



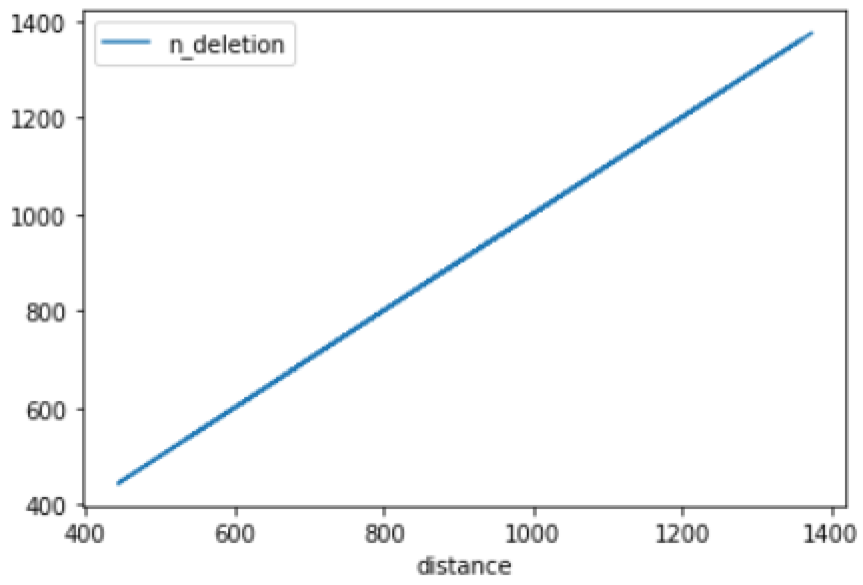
**PLOT#7**:Runtime Vs Number of Deletions
**INSIGHT#7**: Runtime of the algorithms that has used array based queue as back end(BFS_2, A*_2) is far higher than the ones that has used heap priority queue
**INSIGHT#7.1**: Number of deletions do effect runtime in a approximately linear fashion



**PLOT#8**: Runtime(sec) Vs Number of Deletions
**INSIGHT#8**: In general number of deletions does not increases run time so much when heap based priority queue used
**INSIGHT#8.1**: Algorithm used does not differentiate the time taken much as long as heap based priority queue used.

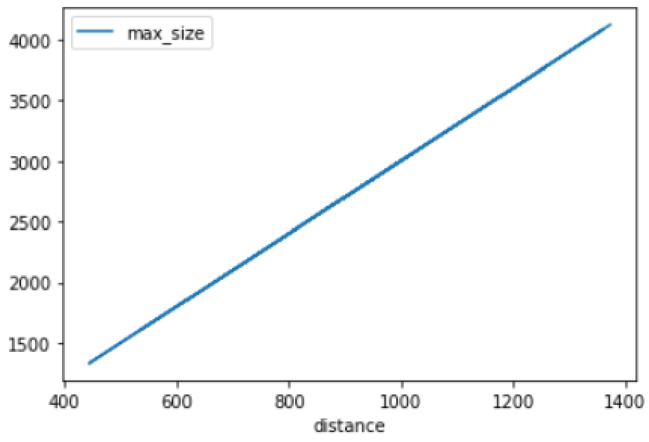


**PLOT#9**:Number of Deletions Vs Runtime(sec) using Array Based Queue
**INSIGHT#9**: The algorithm used does not change the runtime much but rather the number of deletions.
**INSIGHT#9.1**: When array based queue used, it tends to be computationally extensive.



**PLOT#10**:Distance Vs Number of Deletions

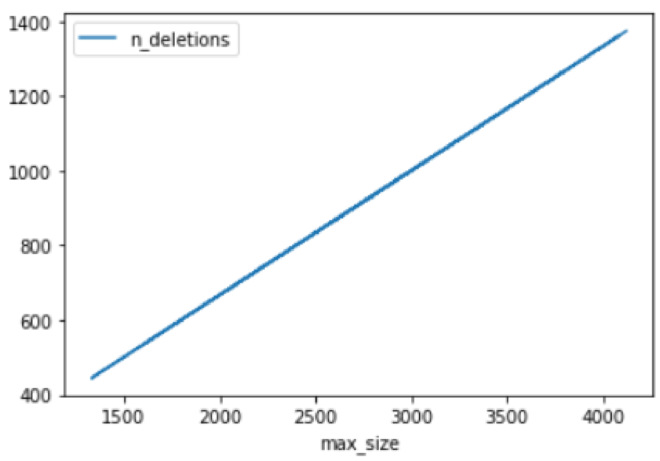
**INSIGHT#10**: In general Distance and Number of Deletions is almost same



**PLOT#11**:Distance Vs Max Queue Size

**INSIGHT#11**: As distance between start and target increase, max queue size is also increasein a linearly fashion

**INSIGHT#11.1**: Maximum queue size used is tend to be 3 fold of distance between starting and target point

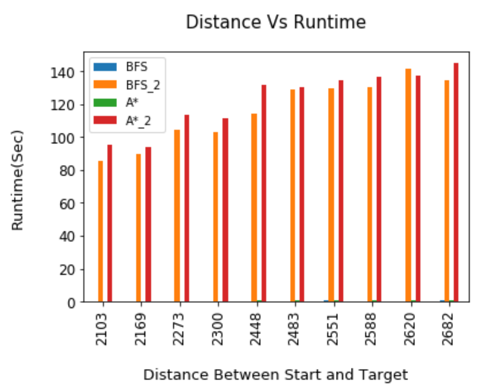


**PLOT#13**:Max Queue Size Vs Number of Deletions

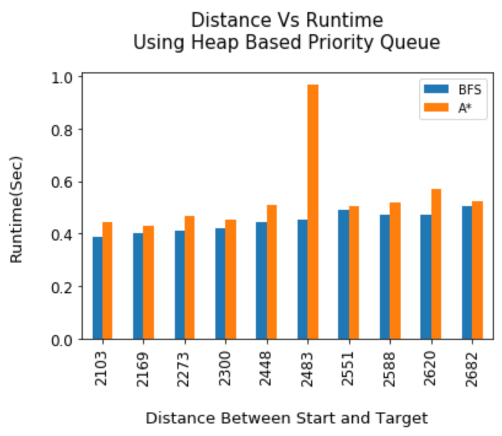
**INSIGHT#13**: As max queue size used increase, number of deletions also increases in a linearly fashion

**INSIGHT#13.1**: Maximum queue size used is tend to be 3 fold of number of deletions

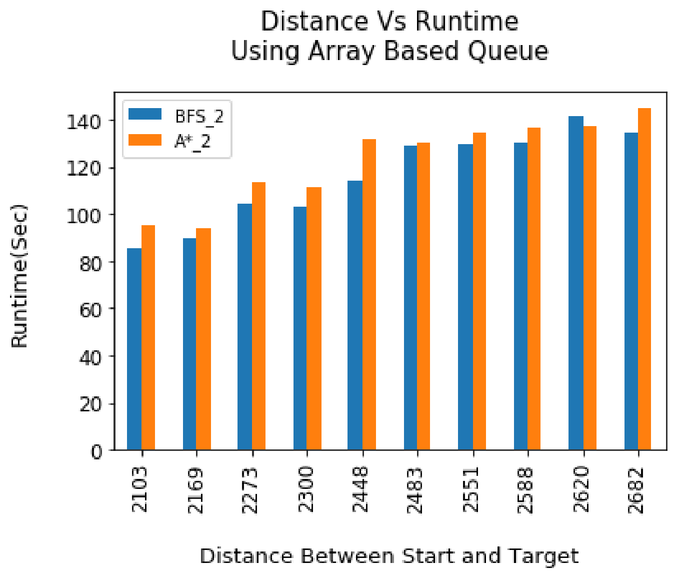
# Case 2: When Distance is So Big



**PLOT#1**:Distance Vs Runtime
**INSIGHT#1**: As distance between start and target increase,the runtime of the algorithms that have been implemented using array based queue(BFS_2, A*_2) outweigh the runtime of the algorithms that have been implemented using heap based priority queue(BFS,A*)
**INSIGHT#1.1**: The runtime of the algorithms that have been implemented using array based queue(BFS_2, A*_2) is so inefficient in compared to the the algorithms that have been implemented using heap based priority queue(BFS,A*)



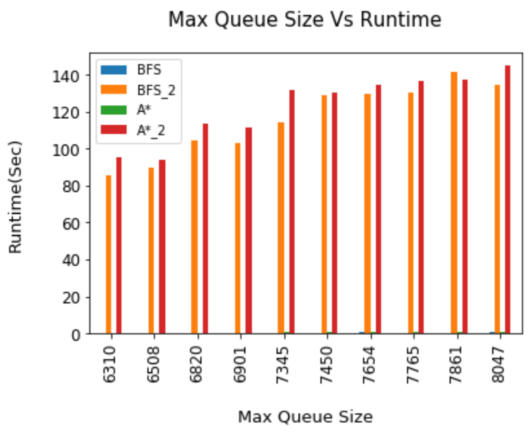
**PLOT#2**:Distance Vs Runtime Using Heap Based Priority Queue
**INSIGHT#2**: When heap based priority queue used, as distance between start and target increase, the runtime does not change radically but rather just small changes



**PLOT#3**:Distance Vs Runtime Using Array Based Queue

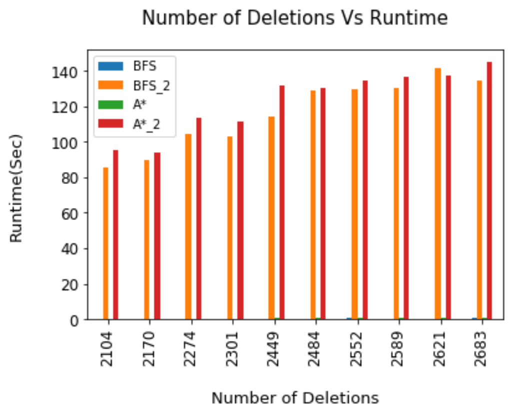
**INSIGHT#3**: Array based queue takes so much time when the distance is high.

**INSIGHT#3.1**: When array based queue used, algorithm chosen does not differentiate the runtime much



**PLOT#4**:Max Queue Size Vs Runtime

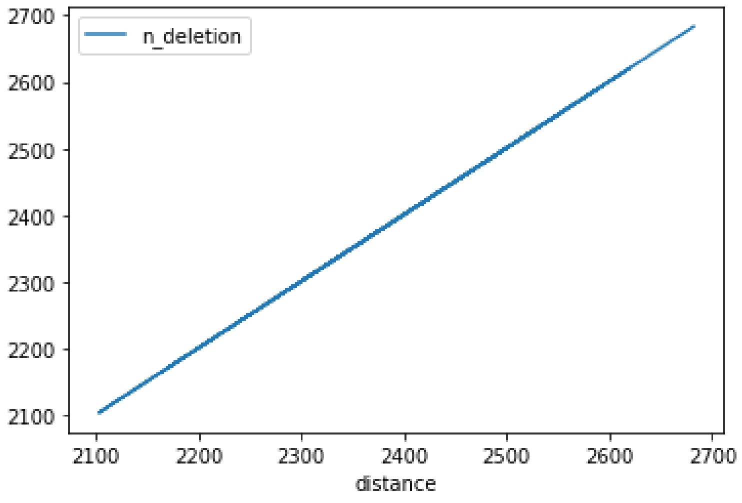
**INSIGHT#4**: As maximum queue size increase, runtime of the algorithms that have been implemented using array based queue becomes so inefficient in compared to the ones that have been implemented using heap based priority queue



**PLOT#7**:Runtime Vs Number of Deletions

**INSIGHT#7**: Runtime of the algorithms that has used array based queue as back end(BFS_2, A*_2) is far higher than the ones that has used heap priority queue

**INSIGHT#7.1**: Number of deletions do effect runtime in a linear fashion with low constant factor



**PLOT#10**:Distance Vs Number of Deletions

**INSIGHT#10**: Distance and Number of Deletions have one-to-one relation