Algorithm PA2 report

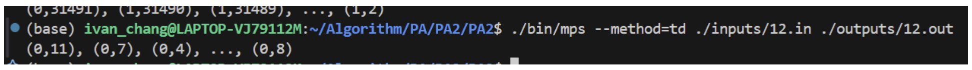
Ivan Chang

November 1, 2025

1 Data Structure

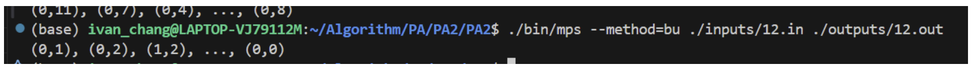
- **vector**: vector is a dynamic array which supports allocating memory dynamically. In my assignment, I use vector to store the dp value.
- **pair**: pair is a data sturcture which can pair two values(whether or not they are the same format) in my assignment, I use it to represent the chords. For chord ij, I store it as (min(i,j), max(i,j))

2 Method Output



```
(0,31491), (1,31490), (1,31489), ..., (1,2)
(base) ivan_chang@LAPTOP-VJ79112M:~/Algorithm/PA/PA2/PA2$ ./bin/mps --method=td ./inputs/12.in ./outputs/12.out
(0,11), (0,7), (0,4), ..., (0,8)
```

Figure 1: Output for td method



```
(0,11), (0,7), (0,4), ..., (0,8)
(base) ivan_chang@LAPTOP-VJ79112M:~/Algorithm/PA/PA2/PA2$ ./bin/mps --method=bu ./inputs/12.in ./outputs/12.out
(0,1), (0,2), (1,2), ..., (0,0)
```

Figure 2: Output for bu method

3 Result Analysis

I have run my code on edaunion server. and iterate through all the test case from 12, 1000, 10000, 20000, 40000, 60000, 80000, 100000, 120000, 180000 Below are the times required and the memory used for each test case.

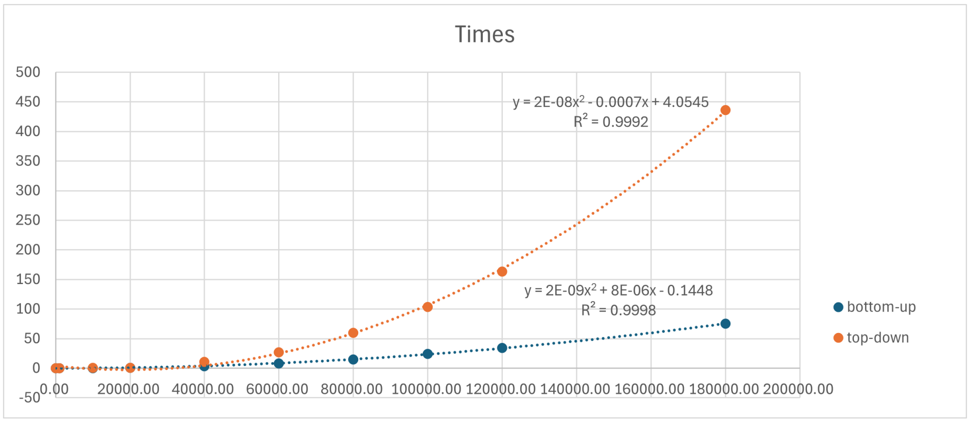


Figure 3: Time vs Input Size

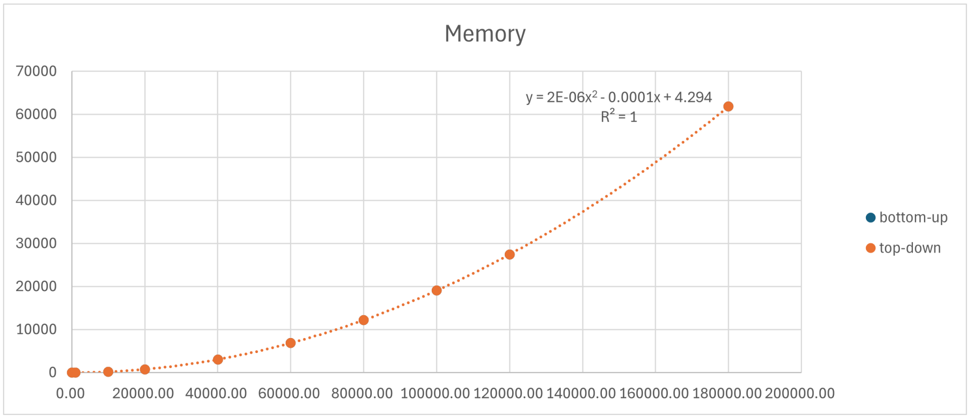


Figure 4: Memory vs Input Size

3.1 discussion

- From figure 2, we can see that the space complexity is $O(n^2)$ since we open a triangular 2D array to store the dp value.
- From figure 1, we can see that the time complexity is also $O(n^2)$, which is expected since we need to fill in the 2D dp array.
- bottom-up method is faster than top-down method. This is because in bottom-up method, we do not have the overhead of recursive function calls. and the cache issue is solved through my adjustment.

4 Other Findings

1. the max input size is 180000. Since each int value takes 4 bytes. We need $180000 * 180000 * 4 \approxeq 129GB$ which is far beyond the memory limit. However, we notice that i ¡= j for all cases. which implys that we only need $\frac{2n*2n+1}{2}$ spaces for storage. It can reduce the size by half.
2. For bottom-up dp, the iteraration order matters. Instead of iterating l to i, we have to iterate from j to i. This is because of the performance of the cache. when we

access memory, the cache will load a block of memory into the cache line. Through my adjestment, the hit rate of cache is improved significantly. So the speed of my bottom-up method is improved.