B11901136 電機四 梁程捷 PA2 Report

The machine used in EDAUnion: (port 40062)

Data Structures

```c
int *chord = malloc(sizeof(int) * n);
bool *is_head = calloc(n, sizeof(bool));
// read the chord data
for(int i = 0; i < N; i++){
    int a, b;
    fscanf(input, "%d %d", &a, &b);
    // printf("Read chord: %d - %d\n", a, b); // Debug
    chord[a] = b; chord[b] = a;
    is_head[a] = true; is_head[b] = false;
}
```

(a) **Array** used to save endpoints of chords

```c
int **M = malloc(sizeof(int*) * (2*N));
for (int i = 0; i < 2*N; i++) {
    M[i] = calloc(2*N - i, sizeof(int));
}
```

(b) Upper half triangle matrix to store the subproblems

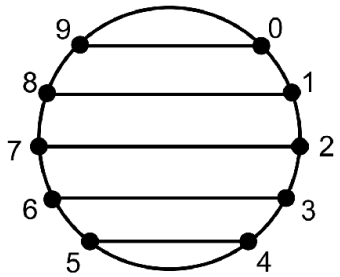
An **array** is used to save the endpoints of the chords as pairs. Another boolean array is used to store the first endpoint of a chord for the sorting purpose afterwards.The reason of not using **vector** in C++ is we already know the size of the array before the algorithm runs, so the unnecessary reallocation of memory by **vector** is avoided.

An upper half triangle array is used to store the subproblems. Special memory mapping is used. We access `M[i][j]` by mapping the coordinates to `M[i][j-i]`.

Discussion of Top-down and Bottom-up approach

The visit sequences of subproblems using the top-down and bottom-up approach for the case (N = 6) are as follows:
Top-Down: (0,11), (0,10), (0,9), ..., (9,9)
Bottom-Up: (0,1), (1,2), (2,3), ..., (0,11)
Top-Down: Begins with the largest subproblem (0,11) and recursively decomposes it into smaller, dependent subproblems like (1,2) and (2,3), solving them on-demand.
Bottom-Up: Systematically solves all subproblems from the smallest (e.g., (0,1)) upwards, using these pre-computed results to efficiently solve larger subproblems like (0,10) and finally (0,11).

Despite the time complexity of both approaches are $O(N^2)$, but in practice, they make a large runtime difference. For example, consider the following case.



(a)

Bottom-up: Solves all subproblems without considering whether the subproblems contribute to the final solution or not.
Top-down: Only solves the required subproblems, which results in solving way less subproblem using bottom-up approach.