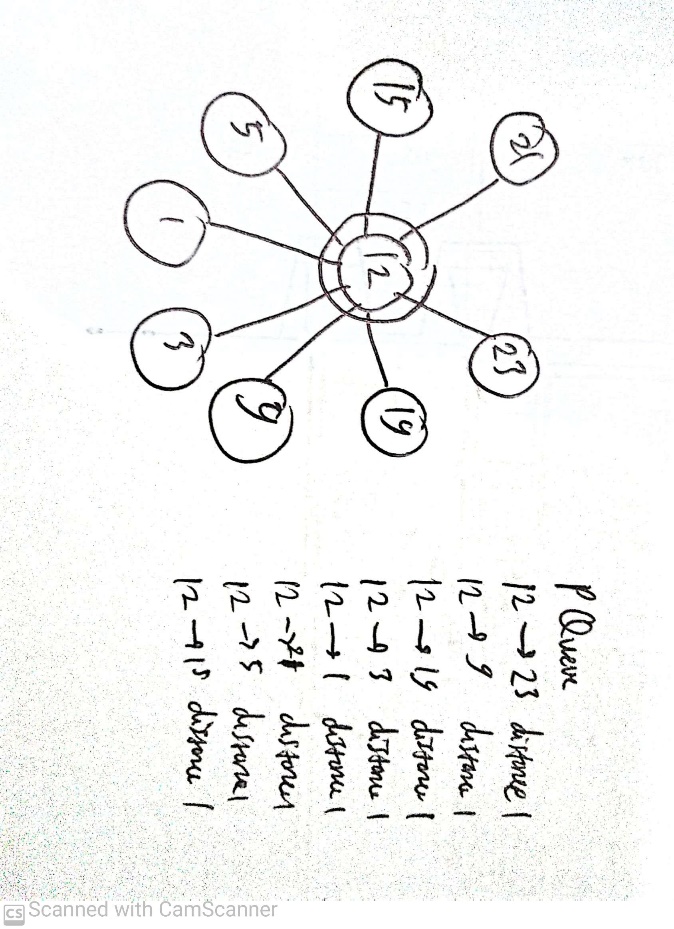
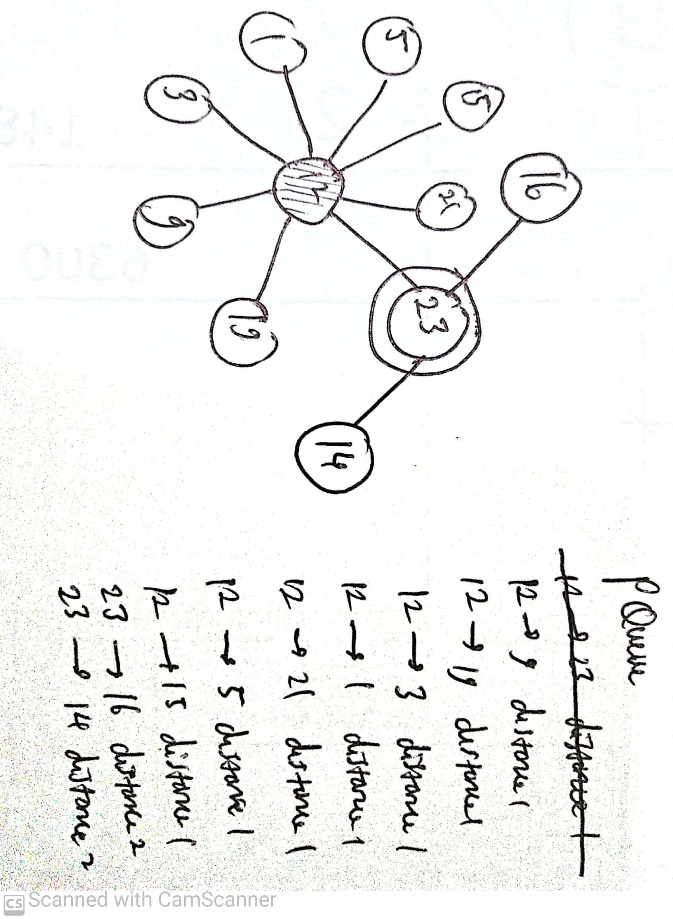
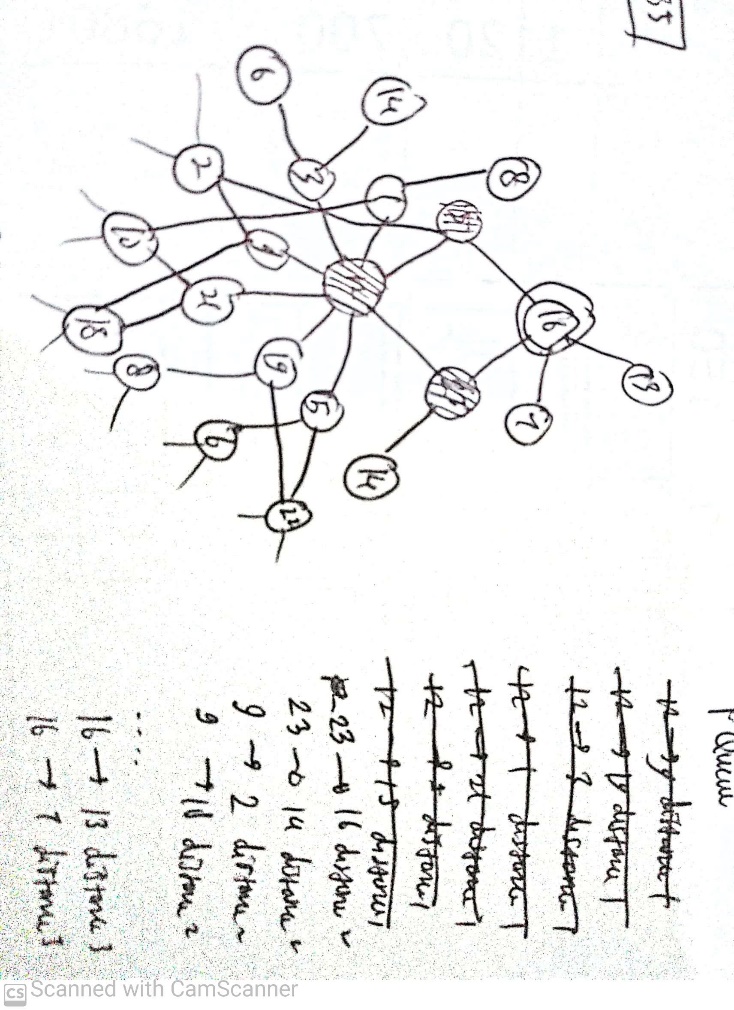
1. Dari antara AA trees,2-3 trees, splay trees,scapegoat trees, dan treaps masing masing ada kelebihan dan kekurangan pada kasus kasus tertentu. Untuk AA trees (red black tree dengan syarat tambahan tidak ada anak left-red)dan treaps (mengandalkan pengurutan random untuk menghindari skewet tree)kedua nya memiliki kelebihan utama dalam mempermudah implementasi dengan tentu mengorbankan perfomance pada kasus kasus tertentu sedangkan 2-3 trees merupakan bentuk sederhana dari B-tree dengan masing masing nodes nya memiliki 2 anak bila satu key nodes dan 3 bila 2 key nodes kemudian splay trees memprioritas kan temporal locality dan scape goattree menghemat space dengan tidak perlu menyimpan faktor penentu pada masing masing nodes dan ada dilakukan perombakan dan penyususan ulang tree. Dari berbagai tree ini diatas saya memili splay tree,splay tree berbeda dengan sifat strictly balanced AVL yang melakukan rotasi setiap insertion, splay tree mengutamakan feature find dimana data yang ingin akses akan di rotasikan diletakkan di root untuk mendapatkan O(1) bila selanjutnya node tersebut akan diakses kembali. Konsep splay tree ini mirip dengan cache pada memory komputer.meski worst case nya O(n) dimana node yang ingin kita akses terletak diujung tree dari node root namun implementasi splay tree ini biasa nya digunakan bila kita ingin mengakses data yang sama secara berulang dan jarang melakukan pencarian node yang berbeda.
2. Daerah pergerakan yang di dapat dikunjungi dapat kita representasikan sebagai node vertex yang saling terhubung dengan edge menjadi sebuah graph. Karena kuda hanya dapat mengunjungi sala satu kemungkinan node dari semua kemungkinan itu dalam satu gerakan maka hubungan antar node dihubungkan ke satu titik utama membentuk tree. Hubungan antar nodes ini dapat kita simpan dalam bentuk matrix(2d array) ataupun linked list dengan property hubungan yang sama. Urutan pengunjungan yang ingin kita lakukan dapat menggunakan konsep stack untuk DFS atau queue untuk BFS. Berikut adalah simulasi sederhana BFS dalam tree. Pertama kali dilakukan pengecekan pada node pertama yang kita kunjungi dan mengisi prioriy queue kita dengan semua kemungkinan nodes yang dapat dikunjungi berdasarkan urutan queue.



Tahap berikut nya dapat terlihat bahwa nodes yang telah kita kunjungi tidak perlu kita tampung lagi hal ini dapat dilakukan dengan menglist node yang telah/tidak perlu kita kunjungi dan melakukan pengcekan bila node yang kita kunjungi sudah terlist atau belum atau menggunakan konsep linked list atau union sets. Saat mengunjungi sebuah node kita menglist semua node yang berhubungan dengan node tersebut pada priority queue kita, menaruh pada list seen kemudian menghapus priority queue teratas.

Setelah beberapa tahap dilakukan terlihat bahwa semua node akan akhir nya dikunjungi hingga tidak tersisa lagi node yang belum di kunjungi.

1. Text editor menggunakan konsep trie (trie.cpp)
2. AVL.pdf, 2-3.pdf, Red-Black.pdf
3. Prim.pdf,kruskal.pdf,Djikstra1.pdf,Djikstra2.pdf