<u>APS Project List</u>

1. **Convex hull and various algorithms for obtaining the 2d convex hull .**
   Description : Convex hull is popular concept from geometry. The objective of this project is to study at least three algorithms for obtaining a convex hull in the 2-d setting and compare the practical efficiency of the algorithms.
   Outcome : Finding convex hull is one of the important task in real life computer vision problems like, autonomous moving car, If the convex hull of a car avoids collision with obstacles then so does the car. Hence the project aims to explore different convex hull algorithms and the improvement over standard algorithms.

2. **Multi-dimensional trie? or Multidimensional suffix tree.. does this exist as of today? Need a 2-d block as a matching or leading prefix.**
   Description : Suffix tree/Trie are most widely used Data structure for pattern searching and string matching. This is an explorative question which deals with exploring whether multidimensional trie/suffix tree exists? If so implement and explain with use cases.
   Outcome : Pattern searching and string matching is one of the major research topic in field of bioinformatics. Project aims to explore how multidimensional trie and suffix tree can be used in those field and demonstrate with example.

3. **Bloom filter and read about graphs stored via bloom filters.**
   Description : Read and understand the Bloom filter data structure and also find at least two applications of the Bloom filter. Implement the two applications and identify the advantages that Bloom filter brings to these applications.
   Outcome : Project aims to learn about probabilistic data structure and their real life application. Bloom Filters is one such data structure. Google Bigtable, Apache HBase and Apache Cassandra, and Postgresql use Bloom filters to reduce the disk lookups for non-existent

rows or columns. Avoiding costly disk lookups considerably increases the performance of a database query operation.

4. **Planar graph APSP and its variations.**
   Description : Study algorithms for all-pairs-shortest-paths on planar graphs including some recent heuristics. Implement one such algorithm and demonstrate the practicality compared to a standard algorithm.
   Outcome : Shortest path algorithms are one among the largely introspected topic in graph theory. Project aims to explore APSP algorithms for planar graph and some heuristics to improve in some real life problems.

5. **Fibonacci heap**
   a) Study the Fibonacci heap data structure and implement the same
   b) Study at least two applications of the data structure to some algorithms such as Kruskal/Prim/Dijkstra as applicable.

6. **Study splay tree and compare wrt AVL tree**
   Study implement and analyse the splay tree, and compare the performance of the splay tree with that of an AVL tree. You have to make automated test cases and show performance difference with respect to the data structures. Any Graphical in-depth analysis would be a bonus.

7. **Study splay tree and compare wrt Red Black tree**
   Study implement and analyse the splay tree, and compare the performance of the splay tree with that of a Red Black tree. You have to make automated test cases and show performance difference with respect to the data structures. Any Graphical in-depth analysis would be a bonus.

8. **Binomial heap with applications and compare wrt AVL and RB-Tree :**
   Implement a binomial heap and compare its performance with AVL and RB-Tree on all major operations. You also have to test your data structure on any of the standard algorithm (Kruskal or Prims). You have to make

automated test cases and show performance difference with respect to the data structures. Any Graphical in-depth analysis would be a bonus.

9. Study your implementation of the **van Emde Boas tree with application to Dijkstra and compare wrt Fibonacci and Binomial.**
Task- Implement a vEB tree and use this to implement Dijkstra. Compare the performance of this Dijkstra with Dijkstra using Fibonacci and Binomial.

10. Study your implementation of the **van Emde Boas tree with application to Dijkstra and compare wrt Red Black Tree and AVL.**
Task- Implement a vEB tree and use this to implement Dijkstra. Compare the performance of this Dijkstra with Dijkstra using Red Black Tree and AVL.

11. Study your implementation of the **van Emde Boas tree with application to Kruskal and compare wrt Fibonacci and Binomial.**
Task- Implement a vEB tree and use this to implement Kruskal. Compare the performance of this Kruskal with Kruskal using Fibonacci and Binomial.

12. Study your implementation of the **van Emde Boas tree with application to Kruskal and compare wrt Union Find and Fibonacci.**
Task- Implement a vEB tree and use this to implement Kruskal. Compare the performance of this Kruskal with Kruskal using Fibonacci and Union Find.

13. Study your implementation of the **van Emde Boas tree with application to Prims and compare wrt Union Find and AVL.**
Task- Implement a vEB tree and use this to implement Prims. Compare the performance of this Prims with Prims using AVL and Union Find.

14. Study your implementation of the **van Emde Boas tree with application to Prims and compare wrt Union Find and RB Tree.**

Task- Implement a vEB tree and use this to implement Prims. Compare the performance of this Prims with Prims using RB Tree and Union Find.

15. Study your implementation of the **van Emde Boas tree with application to Prims and compare wrt RB Tree and AVL.**
Task- Implement a vEB tree and use this to implement Prims. Compare the performance of this Prims with Prims using AVL and RB Tree.

16. Comparison of **Randomized algo and Deterministic algo**. Compare on the basis of accuracy vs performance tradeoff. (Primality).

17. Study your implementation of the **van Emde Boas tree with application to Kruskal and compare wrt Fibonacci and Binomial.**
Task- Implement a vEB tree and use this to implement Kruskal. Compare the performance of this Kruskal with Kruskal using Fibonacci and Binomial.

18. Persistence is a concept in data structures, where we reuse the original state of data structure whenever some update is required. This results in a data structure where we can see the state of data structure before and after the update with minimal use of memory.
Segment trees are interval trees over an array where we can store informations about segments. It allows us to update the information regarding some segment of the array in O(log(n)) time complexity. Your task is to create a **persistent segment tree** where we should be able to get the info of any segment after any update performed.

19. Persistence is a concept in data structures, where we reuse the original state of data structure whenever some update is required. This results in a data structure where we can see the state of data structure before and after the update with minimal use of memory.
Binary Tries are a data structure which allows us to store integers by storing their bits in a trie like format. Your task is to build a **persistent**

binary trie, where we should be able to see numbers in the structure after any update.

20. **Binomial heaps** are data structure which uses binomial trees to perform standard heap operations. In addition to binary heap they provide O(1) merging of two heaps, which is in fact O(N) in binary heap. Your task is to build a binary heap with all the standard heap functions and a merge function which works in O(1) time.

21. Study the **Bitonic sort and the Odd-even sorting algorithms** from either the Cormen book or elsewhere. Implement them and see how they work compared to traditional sorting algorithms like quick sort and merge sort. Benchmark these algorithms on parallel settings as well.

22. **Study the depth of a binary search tree under various operations and proportions of operations and insertion models such as random**. We have seen that the height of a binary search tree depends also on the nature and sequence of operations performed. Study interesting sequences of operations and see how the height varies. Back up with experimental observations.

23. PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. Implement and study **PageRank algorithm**. How can you improve PR in terms of time of execution.

24. Study the analysis, implement the splay tree, and **compare the performance of the splay tree with that of a BST** for different scenarios. Make a detail report of all the benchmarking and case studies.

25. **2-D string matching and pattern identification .**

Study algorithms for this problem and implement at least 3 of them. The Two Dimensional Pattern Matching Problem is defined as follows: Let Σ be an alphabet.

INPUT:
Text array T[1...n,1...n] and pattern array[1..m,1...m].

OUTPUT:
All locations (i,j) in T where there is an occurrence of the pattern, i.e., T[i+k,j+l] = P[k+1,l+1], $0 \leq k,l < m$.

Can refer this:
http://www.sci.brooklyn.cuny.edu/~shoshana/pub/secondExam.pdf

26. **Data structures for sets -- union, intersection, difference with applications.**

Understand how to handle sets, implement your own set and operate on them including operations such as union, intersection, difference, etc.

27. **Dynamic graph algorithm -- matching/DFS/shortest path/Transitive closure**

Read about how to handle edge/vertex insert/deletes in a typical graph algorithm such as matching, DFS, Shortest paths, and transitive closure. Implement all four of problem.

28. **Study your implementation of the van Emde Boas tree with application to Kruskal and compare wrt RB and AVL.**

A Van Emde Boas tree also known as a vEB tree, is a tree data structure which implements an associative array with *n*-bit integer keys. It performs all operations in O(log *n*) time.

Reference to study vEB tree:
http://web.stanford.edu/class/archive/cs/cs166/cs166.1166/lectures/14/Slides14.pdf