|  |  |  |
| --- | --- | --- |
| **Problem** | **Method** | **Comments** |
| **Section 2 - Array** | | |
| Remove Duplicates from sorted array |  |  |
| Remove Duplicates from sorted array,  Allow duplicates up to twice |  |  |
| Search in rotated sorted array | Needs to O(logn). 3-way compare for binary search |  |
| Search in roated sorted array, allow duplicates up to twice |  |  |
| Median of two sorted arrays in O(log(m+n)) |  |  |
| Longest consecutive sequence | Use hashmap |  |
| 2sum |  |  |
| 3sum |  |  |
| 3sum closest |  |  |
| 4sum |  |  |
| Remove element (in-place in O(n)) |  |  |
| Next Permutation |  |  |
| Permutation Sequence |  |  |
| Valid Sodoku |  |  |
| Trapping Rain Water |  |  |
| Rotate Image |  |  |
| Plus One | Use carry, care for details |  |
| Climing stairs | Fibonacci, better use iteration than recursion |  |
| Gray Code |  |  |
| Set Matrix Zeroes (with O(1) space) | Reuse the first row ad first column for flags |  |
|  |  |  |
|  |  |  |
| **Section 3 – Strings** | | |
| Valid Palindrome | transform(s.begin(), s.end(), s.begin(), ::tolower);  ::isalnum(int c); |  |
| Implement strStr() | Brute force, KMP, Boyer-Mooer, Rabin-Karp, Sunday? |  |
| Implement atoi | Lots of details: starting whitespaces, signs, integer range |  |
| Add two binary strings | Reverse the string and use a carry digit. |  |
| Longest Palindromic Substring | DP: Use a 2D array for flagging and caching. |  |
| Regular expresson matching for ‘.’ and ‘\*’ | Recursion or iteration with details. |  |
| Wildcard matching for ‘?’ and ‘\*’ | Recursion or iteration with details. |  |
| Longest common prefix | Simple iteration and checking |  |
| Valid Number | FSM (cool) or strtod (lazy approach) |  |
| Integer to Roman |  |  |
| Roman to Integer |  |  |
| Count and Say |  |  |
| Anagrams | Construct hash from sorted anagrams to original string |  |
| Simplify Path | Corner cases and details here! |  |
| Length of Last Word | find\_if, ::isalpha, distance |  |
|  |  |  |
| **Section 7 - Search** | | |
| Search for a range | std::lower\_bound<vec.begin(), vec.end(), value)  std::upper\_bound<vec.begin(). vec.end(), value)  std::distance(vec.begin(), itr)  std::prev(itr); std::next(itr); | lower\_bound or upper\_bound uses binary search |
| Search insert position | std::lower\_bound(vec.begin(), vec.end(), value) |  |
| Search a 2D matrix | binary search |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Section 10 - DFS** |  |  |
| Parlindrom Paritionning | DFS (recursion) |  |
| Unique paths (in chessboard) | DFS (recursion with cache), or Math formula |  |
| Unique paths II (with obstacles) | DFS (recursion with cache), no easy math solution |  |
| N-Queeen (on N x N chessboard) | DFS with pruning | Backtrack=DFS+Pruning |
| Restore IP address | DFS with pruning |  |
| Combination Sum (select to sum to target) | DFS with pruning |  |
| Generate Parentheses | DFS (simple recursion) |  |
| Sodoku Solver | DFS |  |
| Word Search (from 2D board) |  |  |
|  |  |  |
| **Section 11 – Divide and Conquer** | | |
| Implement pow(x, n) |  |  |
| Implement sqrt(x) |  |  |
|  |  |  |
| **Section 14 – Graphs** | | |
| Clone Graph | Hashmap (std::unordered\_map) for node to node mapping. Do DFS or BFS to clone while maintaining the hash map. |  |