palindrome

number, list

Palindrome Linked List `get_middle, reverse, compare`
Palindrome Number `compute number in reverse`
Find the Closest Palindrome Number `math`
Prime Palindrome `math`
Super Palindromes `math`

permutable string

Longest Palindrome Permutation `even, odd`
Palindrome Permutation I (any solution?) `even, odd`
Palindrome Permutation II (all solution) `Backtracking`

immutable string

Valid Palindrome I `left, right`
Valid Palindrome II (can remove one) `remove either left or right`
Palindrome Pairs `rolling hash`
Longest Palindrome Prefix (Shortest Palindrome with minimum add in front) `DP` `rolling hash`
Longest Palindrome Substring `DP`
Longest Palindrome Subsequence `DP`
Palindrome Partitioning I (all cuts) `DP`
Palindrome Partitioning II (min cut) `DP`

Rolling Hash

```python
# rolling hash
def compute_hash(s, indices, prime=101):
    return sum(ord(s[ind]) * prime ** i for i, ind in enumerate(indices))

def recompute_hash(s, pop, push, hash, length, prime=101):
    return (hash - ord(s[pop])) // prime + ord(s[push]) * prime ** (length-1)

# substring search
def substring_search_brute_force(text, pattern):
    n, m = len(text), len(pattern)
    for i in range(n - m + 1):
        if text[i:i+m] == pattern:  return i
    return -1

def substring_search_rolling_hash(text, pattern):
    n, m = len(text), len(pattern)
    h, hi = compute_hash(pattern, range(m)), compute_hash(text, range(m))
    for i in range(n - m + 1):
        if i > 0: hi = recompute_hash(text, i-1, i+m-1, hi, m)
        if hi == h and text[i:i+m] == pattern:  return i
    return -1
```