

Perfect Squares

Saturday, December 7, 2024 9:15 PM

Updated and Corrected Code:

python

Copy code

class Solution:

```
def numSquares(self, n: int) -> int:
    # Generate all perfect squares less than or equal to n
    memo = []
    i = 1
    while i * i <= n:
        memo.append(i * i)
        i += 1

    # Initialize the visited set and queue for BFS
    visited = set([0]) # Start with 0, as we start from 0
    q = [0] # Queue to store the current sum of squares
    d = 0 # Number of levels (i.e., the number of squares to sum up to n)

    # BFS to find the minimum number of perfect squares summing to n
    while q:
        d += 1
        next_q = []
        for v in q:
            for w in memo:
                temp = v + w
                if temp == n:
                    return d
                if temp < n and temp not in visited:
                    visited.add(temp)
                    next_q.append(temp)
        q = next_q

    return d
```

Explanation:

1. Generate Perfect Squares:

- We generate all perfect squares (1^2 , 2^2 , 3^2 , etc.) that are less than or equal to n . These squares will be used to form sums that can reach n .

2. BFS to Find the Minimum Number of Perfect Squares:

- The idea is to use **Breadth-First Search (BFS)** to explore the possible sums.
- We start with a queue q initialized with 0, and we maintain a visited set to avoid revisiting the same sum repeatedly.
- For each number in q , we try adding each perfect square (w), and if the resulting sum $temp$ equals n , we return the number of steps (d) it took to get there.
- If $temp$ is still less than n and not visited, we add it to the $next_q$ for further exploration.
- After all numbers at the current level are processed, we move to the next level by setting $q = next_q$.

3. Time Complexity:

- **$O(n)$** , since we explore all sums up to n and for each sum, we explore each perfect square. In the worst case, the number of sums is proportional to n and the number of perfect squares is approximately \sqrt{n} .

4. Space Complexity:

- **$O(n)$** , due to the space used by the visited set and the queue q to store intermediate sums.

Example Walkthrough:

Let's say $n = 12$:

- Perfect squares: [1, 4, 9]
- **BFS Level 1:** Start with 0. Possible sums are 1, 4, 9.
- **BFS Level 2:** From 1, we can add 1, 4, 9 to get 2, 5, 10. From 4, we get 5, 8, 13 (but 13 exceeds 12).
- **BFS Level 3:** Continue similarly until we reach 12.

The minimum number of jumps is the level at which we first reach 12.

4o mini

You said:

can you walk through it in detail and give me other examples that are like this problem

ChatGPT said:

ChatGPT

Sure! Let's walk through the algorithm in detail and then I'll provide a couple of other examples that are similar in structure and concept.

Problem Walkthrough

The problem at hand is finding the minimum number of perfect squares that sum up to n . We can break this down step by step using **Breadth-First Search (BFS)**.

The core idea is to explore all possible ways to sum perfect squares to reach the target n . BFS is a good fit here because it explores all possibilities level by level, ensuring that we find the minimum number of perfect squares as soon as possible.

Example 1: $n = 12$

Step 1: Generate Perfect Squares

- First, we generate all perfect squares that are less than or equal to $n = 12$.
 - The perfect squares are [1, 4, 9] because:
 - $1^2 = 1$
 - $2^2 = 4$
 - $3^2 = 9$
 - $4^2 = 16$, but 16 exceeds 12, so we stop here.

Step 2: BFS Initialization

- We initialize:
 - $\text{memo} = [1, 4, 9]$ (the list of perfect squares).
 - $\text{visited} = \{0\}$ (we start from 0, which means no sum yet).
 - $q = [0]$ (queue to store sums).
 - $d = 0$ (the current number of levels in BFS, which represents the number of jumps or steps).

Step 3: BFS Loop

- **Level 1 (Initial state):**
 - We start by exploring from 0 (the sum we've reached so far).
 - We add each perfect square from memo to 0:
 - $0 + 1 = 1$
 - $0 + 4 = 4$
 - $0 + 9 = 9$
 - Now, $q = [1, 4, 9]$ and $\text{visited} = \{0, 1, 4, 9\}$.
- **Level 2:**

- We now explore from 1, 4, and 9:
 - From 1, we can add:
 - $1 + 1 = 2$
 - $1 + 4 = 5$
 - $1 + 9 = 10$
 - From 4, we can add:
 - $4 + 1 = 5$ (but it's already visited)
 - $4 + 4 = 8$
 - $4 + 9 = 13$ (but it's greater than 12)
 - From 9, we can add:
 - $9 + 1 = 10$ (already visited)
 - $9 + 4 = 13$ (greater than 12)
 - $9 + 9 = 18$ (greater than 12)
- Now, $q = [2, 5, 8]$ and $visited = \{0, 1, 2, 4, 5, 8, 9\}$.
- **Level 3:**
 - We now explore from 2, 5, and 8:
 - From 2, we can add:
 - $2 + 1 = 3$
 - $2 + 4 = 6$
 - $2 + 9 = 11$
 - From 5, we can add:
 - $5 + 1 = 6$ (already visited)
 - $5 + 4 = 9$ (already visited)
 - $5 + 9 = 14$ (greater than 12)
 - From 8, we can add:
 - $8 + 1 = 9$ (already visited)
 - $8 + 4 = 12$ (success! We found 12).
 - Now, we found 12 at this level, so we return $d = 3$.

Final Answer:

- The minimum number of perfect squares that sum up to 12 is **3** (since $12 = 9 + 1 + 1$).

Example 2: $n = 13$

Step 1: Generate Perfect Squares

- The perfect squares are $[1, 4, 9]$.

Step 2: BFS Initialization

- $memo = [1, 4, 9]$
- $visited = \{0\}$
- $q = [0]$
- $d = 0$

Step 3: BFS Loop

- **Level 1:**
 - We add 1, 4, and 9 to 0:
 - $0 + 1 = 1$
 - $0 + 4 = 4$
 - $0 + 9 = 9$
 - Now, $q = [1, 4, 9]$ and $visited = \{0, 1, 4, 9\}$.
- **Level 2:**
 - From 1, 4, and 9:
 - From 1, add:
 - $1 + 1 = 2$
 - $1 + 4 = 5$
 - $1 + 9 = 10$
 - From 4, add:

- $4 + 1 = 5$ (already visited)
- $4 + 4 = 8$
- $4 + 9 = 13$ (success! We found 13).
- Now, we found 13 at this level, so we return $d = 2$.

Final Answer:

- The minimum number of perfect squares that sum up to 13 is **2** (since $13 = 9 + 4$).

Example 3: $n = 6$

Step 1: Generate Perfect Squares

- The perfect squares are $[1, 4]$.

Step 2: BFS Initialization

- $memo = [1, 4]$
- $visited = \{0\}$
- $q = [0]$
- $d = 0$

Step 3: BFS Loop

- **Level 1:**
 - We add 1 and 4 to 0:
 - $0 + 1 = 1$
 - $0 + 4 = 4$
 - Now, $q = [1, 4]$ and $visited = \{0, 1, 4\}$.
- **Level 2:**
 - From 1 and 4:
 - From 1, add:
 - $1 + 1 = 2$
 - $1 + 4 = 5$
 - From 4, add:
 - $4 + 1 = 5$ (already visited)
 - $4 + 4 = 8$ (greater than 6)
 - Now, $q = [2, 5]$ and $visited = \{0, 1, 2, 4, 5\}$.
- **Level 3:**
 - From 2 and 5:
 - From 2, add:
 - $2 + 1 = 3$
 - $2 + 4 = 6$ (success! We found 6).
 - Now, we found 6 at this level, so we return $d = 3$.

Final Answer:

- The minimum number of perfect squares that sum up to 6 is **3** (since $6 = 4 + 1 + 1$).

Similar Problems:

1. Minimum Number of Coins for Change:

- **Problem:** Given a list of coin denominations and a target amount, find the minimum number of coins needed to make the target amount.
- **Approach:** Use dynamic programming or BFS, similar to the BFS approach used for the minimum number of perfect squares.

2. Word Ladder:

- **Problem:** Given a start word, end word, and a dictionary of words, find the shortest transformation sequence from the start word to the end word such that only one letter changes at a time, and each transformed word must exist in the dictionary.
- **Approach:** Use BFS to explore all possible transformations.

3. Unbounded Knapsack (Subset Sum):

- **Problem:** Given a set of items with values and weights, and a knapsack with a weight capacity, find the maximum value that can be carried without exceeding the weight.
- **Approach:** Use dynamic programming to build up the solution, which is similar to the BFS-

like approach.

Each of these problems involves exploring combinations of smaller steps (whether it's numbers or words) to achieve a larger goal, and can be solved with BFS, dynamic programming, or similar techniques.