2591. Distribute Money to Maximum Children

Easy Greedy Math

Problem Description

You have a certain amount of money that you need to distribute to a number of children such that three conditions are met:

1. All the money must be distributed: You cannot have any money left once you've met the distribution requirements.
2. Everyone must receive at least 1 dollar: No child should receive less than a dollar.
3. Nobody receives 4 dollars: Due to unspecified reasons, giving exactly 4 dollars to any child is not allowed.

The challenge here is to find out the maximum number of children who can receive exactly 8 dollars, within the constraints provided. If you cannot distribute the money as per the rules, then the answer should be -1.

Intuition

The strategy to solve this problem uses a case analysis approach. Here's the intuition behind arriving at the correct answer:

Firstly, it's clear that if you have less money than the number of children, you can't distribute the money since everyone needs to get at least one dollar. That case directly leads to a return value of -1.

Next, consider the possibility that you have more money than necessary to give each child 8 dollars. If this is the situation, you will distribute 8 dollars to `children−1` kids, and the last child will get the remaining money, granting that it's not equal to 4. Because we want to maximize the children getting exactly 8 dollars, we can safely say that in this case, `children−1` will be receiving 8 dollars each.

Another case to look at is where the total money is exactly 4 dollars less than 8 times the number of children, i.e., `money = 8 * children − 4`. Here, `children−2` kids can get 8 dollars because we can't give out the 4 dollars as per the rules, and we'd be left with 12 dollars to split between the remaining two children in some way that doesn't involve giving 4 dollars to either.

Finally, for other scenarios, you find the maximum number of children that can receive 8 dollars by assuming `x` children receive it. The leftover money would be `money − 8x`. As long as the remaining amount is at least equal to the number of children that are left (`children − x`), the distribution is possible. To find the largest possible `x`, you can solve for `x` using the expression `x <= (money − children)/7`, which ensures each of the remaining children gets at least 1 dollar without anyone getting exactly 4 dollars.

By analyzing these cases and creating conditions within the solution code, we cover all possibilities and can thereby calculate the maximum number of children that can receive exactly 8 dollars complying with the distribution rules.

Solution Approach

The implementation of the solution uses a series of checks to handle the different cases that arise from the problem's constraints. Here's a breakdown of how each part of the solution corresponds to the scenarios mentioned:

1. Case of insufficient money (`money < children`): The implementation directly returns `−1` if `money` is less than `children` since it's not possible to distribute less money than the number of recipients if each must get at least $1.
2. Case of more money than needed for 8 dollars each (`money > 8 * children`): The code returns `children − 1` since you can give 8 dollars to each child except for one. The last child will get the remaining amount, which is guaranteed to be more than 8 dollars (and since no child can receive exactly 4 dollars, the last child cannot end up with exactly 4 dollars, and this distribution is valid).
3. Case of `money` being exactly 4 dollars less (`money == 8 * children − 4`): In this particular case, the solution returns `children − 2` since you can give 8 dollars to `children − 2` kids. The last two children will have to share the remaining 12 dollars in a way that avoids giving either of them exactly 4 dollars.
4. The general case: If none of the above specific conditions are met, the code uses the formula `x <= (money − children)/7` to determine the maximum number of children who can receive exactly 8 dollars. This comes from ensuring that the remaining money after giving `x` children 8 dollars each (`money − 8x`) must be at least as much as the number of children who still need to receive money (`children − x`). Consequently, the solution finds the integer division of `(money − children) // 7` to get the maximum number of children that can get 8 dollars each. Integer division is used here since we are dealing with whole dollars and whole numbers of children.

Each of these steps uses basic arithmetic and control structures — no complex data structures, algorithms, or patterns are needed. This is because of the problem's constraints and the nature of the resource being distributed (money), which can be divided into integer amounts among the children. The implementation is straightforward and mainly revolves around the application of mathematical logic to the problem's rules.

Example Walkthrough

Let's use a small example to illustrate the solution approach. Suppose you have 52 dollars to distribute among 5 children.

First, you test if the total money is less than the number of children, which is not the case here, as 52 dollars is more than enough to give each child at least 1 dollar.

Next, you test if the money is more than enough to give every child 8 dollars each (`money > 8 * children`). In this case, (52 > 8 \times 5) (which is 40) is true. Following the solution approach, you can distribute 8 dollars to each of 4 children (which totals to 32), and the last child will get the remaining (52 - 32 = 20) dollars. This satisfies the condition that no child receives exactly 4 dollars, as the last child receives much more than that.

Then, you also have to check for the case where you have just 4 dollars less than what you would need to give everyone 8 dollars (`money == 8 * children − 4`). For our example, (52 \neq 8 \times 5 - 4) (which is 36), so this is not applicable.

Finally, if none of the specific conditions apply, which is not the case here since we found our solution in the second step, you would use the general formula (x \leq (money - children) / 7) to determine the maximum number of children you can give 8 dollars to. In this scenario, the calculation would be (x \leq (52 - 5) / 7), which simplifies to (x \leq 47 / 7), and finally to (x \leq 6). However, since this formula gives you a number higher than the number of children, it simply means you can cover giving 8 dollars to each child, and then some, which we already determined.

In summary, for our example with 52 dollars and 5 children, based on the solution approach, we would be able to meet the conditions and give out 8 dollars each to 4 of the children, while the fifth child would receive the remaining 20 dollars. Thus, the answer to our problem would be 4 children receiving exactly 8 dollars each.

Solution Implementation

Python

```python
class Solution:
    def dist_money(self, money: int, children: int) -> int:
        # If there is less money than children, distribution isn't possible.
        if money < children:
            return -1

        # If money exceeds 8 times the number of children, each child can receive
        # at least one coin and the solution approaches the number of children minus 1.
        # This is because we can distribute 7 coins without forming an octagon.
        if money > 8 * children:
            return children - 1

        # If the amount of money is exactly 4 less than 8 times the number of children,
        # we face a specific case where we cannot form a complete last octagon and
        # therefore, must subtract an additional child from the normally expected count.
        if money == 8 * children - 4:
            return children - 2

        # In cases where the constraints above aren't met, the number of children that
        # need to be skipped (to avoid creating octagons) can be calculated by dividing
        # the excess money over a 1-to-1 distribution by 7. This ensures that the distribution
        # does not allow for an octagon's worth of coins (8) to any child.
        return (money - children) // 7
```

Java

```java
class Solution {

    // Method to distribute money among children according to a specific rule
    public int distMoney(int money, int children) {

        // If there is less money than the number of children, distribution is impossible
        if (money < children) {
            return -1;
        }

        // If there is more than 8 times the money compared to the children, max result is achieved
        if (money > 8 * children) {
            return children - 1; // Return the maximum number of unique amounts
        }

        // Specific condition check when money equals to a certain factor
        if (money == 8 * children - 4) {
            return children - 2; // Adjust return value for this specific case
        }

        // General calculation for other cases according to the given rules
        // The formula calculates the max number of unique amounts that can be distributed
        return (money - children) / 7;
    }
}
```

C++

```cpp
class Solution {
public:
    int distMoney(int money, int numChildren) {
        // If there is not enough money to give each child at least one unit, return -1
        if (money < numChildren) {
            return -1;
        }

        // If there is more than 8 times the money compared to the number of children,
        // it means each child can receive at least 8 units, and we return the number of leftover units.
        if (money > 8 * numChildren) {
            return numChildren - 1;
        }

        // If the money is exactly 4 units less than 8 times the number of children,
        // we must leave out two children to meet the condition as closely as possible.
        if (money == 8 * numChildren - 4) {
            return numChildren - 2;
        }

        // In other cases, distribute the exceeding amount of money evenly minus the number of children,
        // and the result is how much more each child gets, one by one, until the excess is exhausted.
        return (money - numChildren) / 7;
    }
};
```

TypeScript

```typescript
/**
 * Calculate the distribution of money among children.
 *
 * @param {number} money - The total amount of money to be distributed.
 * @param {number} children - The number of children to distribute the money to.
 * @returns {number} The number of children who receive more than the basic amount, or -1 if distribution is not possible.
 */
function distMoney(money: number, children: number): number {
    // If there isn't enough money to give each child at least $1, return -1 (failure to distribute).
    if (money < children) {
        return -1;
    }

    // If the amount of money is more than eight times the number of children,
    // every child can get more than the base amount, except one.
    if (money > 8 * children) {
        return children - 1;
    }

    // Specific case: when the money is equal to 8 times the number of children minus 4,
    // this implies two children will receive less than the others.
    if (money === 8 * children - 4) {
        return children - 2;
    }

    // In other cases, calculate the number of children who can receive $1 more than the base amount.
    // This is based on dividing the excess money after giving each child $1 by 7.
    return Math.floor((money - children) / 7);
}
```


```python
class Solution:
    def dist_money(self, money: int, children: int) -> int:
        # If there is less money than children, distribution isn't possible.
        if money < children:
            return -1

        # If money exceeds 8 times the number of children, each child can receive
        # at least one coin and the solution approaches the number of children minus 1.
        # This is because we can distribute 7 coins without forming an octagon.
        if money > 8 * children:
            return children - 1

        # If the amount of money is exactly 4 less than 8 times the number of children,
        # we face a specific case where we cannot form a complete last octagon and
        # therefore, must subtract an additional child from the normally expected count.
        if money == 8 * children - 4:
            return children - 2

        # In cases where the constraints above aren't met, the number of children that
        # need to be skipped (to avoid creating octagons) can be calculated by dividing
        # the excess money over a 1-to-1 distribution by 7. This ensures that the distribution
        # does not allow for an octagon's worth of coins (8) to any child.
        return (money - children) // 7
```


Time and Space Complexity

The time complexity of this function is `O(1)` because the operations it performs (comparisons, arithmetical operations, and a division) have constant time complexity and do not depend on the size of the input variables `money` or `children`. The function executes a fixed number of operations regardless of the input values, leading to a constant time complexity.

The space complexity of the function is also `O(1)` because the function does not allocate any additional space that grows with the size of the inputs. It uses a fixed amount of space for a few variables needed for its calculations, which does not change based on the input.