Coin Change

```
class Solution {
public:
    int coinChange(vector<int> coins, int amount) {
      if (amount < 0) return -1;
      if (amount == 0) return 0;

      vector<int> awns(amount+1,amount+1);
      awns[0] = 0;
      for (int i = 1;i<=amount;i++){
        for(int j : coins){
          if (j <= i){
            awns[i] = min(awns[i],awns[i - j] + 1);
          }
        }
      }

      if (awns[amount]==amount+1)awns[amount] = -1;

      return awns[amount];
    }
};
```

- when you know the index using a array is faster then hash map look up
- optimized but with hash map

```
class Solution {
public:
    int coinChange(vector<int> coins, int amount) {
      if (amount == 0) return 0;
      if (amount < 0) return -1;
      struct my { int value=1e9; };
      unordered_map<int,struct my> map;
      map[0].value = 0;
```

```
      for (int i = 1;i<=amount;i++){
        int coin = 0;
        for(int j = 0;j<coins.size();j++){
          if (coins[j] <= i){
            map[i].value = min(map[i].value,map[i - coins[j]].value + 1);
          }
        }
      }

      if (map[amount].value==1e9)map[amount].value = -1;

      return map[amount].value;
    }
};
```

- Unoptimized

```
class Solution {
public:
    int coinChange(vector<int> coins, int amount) {
      unordered_map<int,int> map;
      map[0] = 0;
      for (int i = 1;i<=amount;i++){
        int coin = 0;
        for(int j = 0;j<coins.size();j++){
          int calc = i - coins[j];
          if (calc >=0){
            if (map.count(calc)){
              coin = map[calc] + 1;
              if(map.count(i)){
                map[i] = min(map[i],coin);
              }else map[i] = coin;
            }
          }
        }
      }

      if (map[amount]==0 && amount > 0)map[amount] = -1;
```

```
      return map[amount];
    }
};
```