78. Subsets

Medium 10227 157 Add to List Share

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]
```

Example 2:

```
Input: nums = [0]
Output: [[],[0]]
```

Constraints:

- `1 <= nums.length <= 10`
- `-10 <= nums[i] <= 10`
- All the numbers of `nums` are **unique**.

By using bit multiplication.

```
bool checkBit(int i, int j) {
    return (i>>j) & 1;
}

vector<vector<int>> subsets(vector<int>& nums) {
    int n = nums.size(), p = 1<<n;
    vector<vector<int>> result(p);

    for(int i=0; i<p; i++) {
        for(int j=0; j<n; j++) {
            if(checkBit(i,j)) {
                result[i].push_back(nums[j]);
            }
        }
    }
    return result;
}
```

By using backtracking.

```
vector<vector<int>> allSubsets;
void generate(vector<int> &subset, int i, vector<int>& nums) {

    // base condition:
    if(i == nums.size()) {
        allSubsets.push_back(subset);
        return;
    }

    // don't take ith elements
    generate(subset, i+1, nums);

    // take ith elements
    subset.push_back(nums[i]);
    generate(subset, i+1, nums);
    subset.pop_back();
}

vector<vector<int>> subsets(vector<int>& nums) {
    vector<int> subset;
    generate(subset, 0, nums);
    return allSubsets;
}
```

#100daysofDSA





/rvislive

Rakesh Vishwakarma