C++ Backtracking Templates - Pick / Not Pick Variants

1. Basic Subset Sum (Pick / Not Pick)

```
void backtrack(int i, vector<int> &ds, int sum, const vector<int> &arr, int targetSum) {
    if (i == arr.size()) {
        if (sum == targetSum) {
            for (int x : ds) cout << x << " ";
            cout << endl;
        }
        return;
    }

    ds.push_back(arr[i]);
    sum += arr[i];
    backtrack(i + 1, ds, sum, arr, targetSum);
    ds.pop_back();
    sum -= arr[i];

    backtrack(i + 1, ds, sum, arr, targetSum);
}
```

2. String Subsequences (Pick / Not Pick)

```
void backtrack(int i, string &ds, const string &s) {
    if (i == s.length()) {
        cout << ds << endl;
        return;
    }

    // pick
    ds.push_back(s[i]);
    backtrack(i + 1, ds, s);
    ds.pop_back();

    // not pick
    backtrack(i + 1, ds, s);
}
```

3. Permutations (Distinct Elements)

```
void backtrack(vector<int> &nums, vector<bool> &used, vector<int> &ds) {
    if (ds.size() == nums.size()) {
        for (int x : ds) cout << x << " ";
        cout << endl;
        return;
    }

    for (int i = 0; i < nums.size(); ++i) {
        if (!used[i]) {
            used[i] = true;
            ds.push_back(nums[i]);
```

```
            backtrack(nums, used, ds);
            ds.pop_back();
            used[i] = false;
        }
    }
}
```

4. Permutations (With Duplicates)

```
void backtrack(vector<int> &nums, vector<bool> &used, vector<int> &ds) {
    if (ds.size() == nums.size()) {
        for (int x : ds) cout << x << " ";
        cout << endl;
        return;
    }

    for (int i = 0; i < nums.size(); ++i) {
        if (used[i] || (i > 0 && nums[i] == nums[i-1] && !used[i-1])) continue;
        used[i] = true;
        ds.push_back(nums[i]);
        backtrack(nums, used, ds);
        ds.pop_back();
        used[i] = false;
    }
}
```