Pairs with specific difference

> 对数组元素进行配对，两两之间元素差值必须小于K，且最终所有配对的元素之和要求最大

注意到，一旦数组完成了排序，那么对于arr[i]，假设其既可以与arr[i - 1]进行配对，也可以与arr[i - 2]配对，由于数组有序，arr[i]与arr[i - 1]配对之后的配对和是更大的，所以有如下的动态规划方法：

1. 首先对数组进行排序，令dp[i]代表前i个元素所能获得的最大的配对和
2. 对于dp[i]有（i ≥ 2）
    1. arr[i]与arr[i - 1]进行配对，则dp[i] = dp[i - 2] + arr[i] + arr[i - 1]
    2. arr[i]不与arr[i - 1]进行配对，则dp[i] = dp[i - 1]
3. 对于 i < 2的情况：dp[i] = max(dp[i], arr[i] + arr[i - 1])

```
#include <bits/stdc++.h>

using namespace std;

int sumOfPairs(vector<int> &arr, int &N, int &K) {
    sort(arr.begin(), arr.end());
    vector<int> dp(N);
    dp[0] = 0;
    for (int i = 1; i < N; ++i) {
        dp[i] = dp[i - 1];
        if (arr[i] - arr[i - 1] < K) {
            i >= 2 ? dp[i] = max(dp[i], dp[i - 2] + arr[i] + arr[i - 1]) : dp[i] = max(dp[i], arr[i] + arr[i - 1]);
        }
    }
    return dp[N - 1];
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, K;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) {
            scanf("%d", &arr[i]);
        }
        scanf("%d", &K);
        printf("%d\n", sumOfPairs(arr, N, K));
    }
    return 0;

}
```