K. STICKERS

PROBLEM DESCRIPTION

給兩列總共 2n 個數字，在上面挑選一些不相臨的數字使得其總和為最大。

SOLUTION TECHINQUES

動態規劃

SOLUTION SKETCHES

我們對於每一行 (column) 做討論，其實只有三種狀態：選上面的數字、下面的數字、都不選，由此我們可以列出以下遞迴轉移式：

假設 dp[i][0] 為在 i 行不選最大的總和、dp[i][1] 為選上面的數字、dp[i][2] 為選下面的數字

dp[i][0] = max(dp[I – 1][0], dp[I – 1][1], dp[I – 1][2]);

dp[i][1] = max(dp[I – 1][0], dp[I – 1][2]) + a[i];

dp[i][2] = max(dp[I – 1][0], dp[I – 1][1]) + b[i];

- ⇨ 最後的答案為 max(a[n][0], a[n][1], a[n][2])

TIME COMPLEXITY

每筆測資 O(n)

SOLUTION PROGRAM FOR REFERENCE

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>

using namespace std;

const int N = 1e5 + 2;

int a[N], b[N];
int dp[N][3];

int main() {
    int i, tt, n;
    scanf("%d", &tt);
    while (tt--) {
        scanf("%d", &n);
        for (i = 0; i < n; i++) scanf("%d", &a[i]);
        for (i = 0; i < n; i++) scanf("%d", &b[i]);
        dp[0][0] = 0;
        dp[0][1] = a[0];
        dp[0][2] = b[0];
        for (i = 1; i < n; i++) {
            dp[i][0] = max(max(dp[i - 1][1], dp[i - 1][2]), dp[i
- 1][0]);
            dp[i][1] = max(dp[i - 1][0], dp[i - 1][2]) + a[i];
            dp[i][2] = max(dp[i - 1][0], dp[i - 1][1]) + b[i];
        }
```

```
        int ans = max(dp[n - 1][1], dp[n - 1][2]);
        ans = max(ans, dp[n - 1][0]);
        printf("%d\n", ans);
    }
    return 0;
}
```