АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ИГРА

Първо, за да можем да сметнем бързо сумата от стойностите на чиповете за даден интервал си въвеждаме масив sum[]. Нека sum[i] = a[0] + a[1] + .. + a[i]. Така сумата за даден интервал [i; j] можем да намерим, като от sum[j] извадим sum[i – 1]. Нека cost (i, j) връща sum[j] – sum[i – 1]. Тази функция ще ползваме по-късно в нашето решение.

Ако имаме една редица от чипове, можем да изберем да вземем най-левия или най-десния и избираме този, който ще ни донесе повече чипове в края на играта. Нека функцията get (i, j) пресмята какви са максималните точки, които ще спечели този, който е на ход, ако играят с подредицата в интервала [i; j]. Ако в момента играем в този интервал и сме ние на ход, можем да вземем i-тия чип. Тогава точките, които ще направи нашия противник ще бъдат get (i + 1, j). Тъй като сумата от чиповете в интервала е cost (i, j), бързо можем да сметнем, че нашите точки тогава ще бъдат cost (i, j) – get (i + 1, j). Аналогично, ако вземем j-тия елемент, точките ни ще бъдат cost (i, j) – get (i, j – 1). Т.е. get (i, j) = max { cost (i, j) – get (i + 1, j), cost (i, j) – get (i, j – 1) }. Знаейки, че get (i, i) = a[i], слагаме дъно на горната рекурсия.

Но при тези ограничения за N, е неоходимо дадения алгоритъм да се оптимизира, тъй като иначе ще работи много бавно. Това лесно става с динамично програмиране. За да не пресмятаме всеки път, когато ни се налага, стойноста на get (i, j) наново, въвеждаме масив dp[N][N]. В dp[i][j], ще се пази и връща по-бързо стойността на функцията, ако тя е била сметната преди това. Така сложността на алгоритъма става квадратична.

Кодът на програмата:

```
#include <iostream>
#include <stdio.h>

using namespace std;

int N, sum[128], dp[128][128], a[128];

int cost (int i, int j) {
   if (i == 0) return sum[j];
   return sum[j] - sum[i - 1];
}
int get(int i, int j)
{
   if (dp[i][j]) return dp[i][j];

   if (i == j)
      return a[i];

   dp[i][j] = max(cost (i, j) - get(i + 1, j), cost (i, j) - get(i, j - 1));

   return dp[i][j];
}
```

```
int main()
{
    scanf("%i%i", &N, &a[0]), sum[0] = a[0];

    for (int i = 1; i < N; i++)
        scanf("%i", a + i), sum[i] = a[i] + sum[i - 1];

    get (0, N - 1);

    printf("%i %i\n", dp[0][N - 1], sum[N - 1] - dp[0][N - 1]);
}
```