Câu 3. Tên bài: **TOWER.CPP**

+ Nhận xét: Đây là bài toán tìm dãy con không tăng theo R và có tổng các H là lớn nhất.

+ Gọi $F(x)$ là độ cao lớn nhất của tòa tháp xếp được với khối có x là đường kính hình trụ xếp cao nhất . Gọi $Rmax$ là giá trị đường kính lớn nhất của các hình trụ. Với mỗi khối trụ thứ i, ta sẽ duyệt lại 1 vòng các đường kính từ Ri đến $Rmax$.

Ta thiết lập được công thức truy hồi

$F(Ri) = max(F(Ri), F(j) + Hi)$. Với mọi j thỏa mãn $Ri \le j \le Rmax$.

Do đó, ta cần khởi tạo với mọi i từ 1 đến N thì $F(Ri) = Hi$, còn các F còn lại thì khởi tạo bằng 0.

Kết quả là $max\ (F(Ri))$ với mọi i chạy từ 1 đến N.

Độ phức tạp của thuật toán là $O(N*Rmax) \approx 500 * 10^5 = 5.10^7$

Câu 4. Tên bài: **STADIUM.CPP**

+ Lưu ý ở câu này, chúng ta muốn sử dụng quy hoạch động thì sẽ phải sắp xếp các đội theo 1 chỉ số nào đó để dễ dàng quản lí và xây dựng trạng thái.

+ Nếu sắp xếp các thời gian sử dụng sân theo thời điểm kết thúc, ta sẽ đưa được về bài toán **tìm dãy con có tổng lớn nhất**. Bài toán này là biến thể của bài toán tìm dãy con tăng dài nhất.

Ta gọi $f(i)$ là số tiền lớn nhất ông chủ sân có được khi đến đội thứ i được thuê. Ta có khi xét đến đội i, với $1 \le j < i$, thì nếu $bj \le ai$ thì $f(i) = max\ (f(i), f(j) + ci)$.

Kết quả bài toán là max của các $f(i)$. với mọi i chạy từ 1 đến n

Độ phức tạp của thuật toán là $O(n^2)$.

Câu 5. Tên bài: **EXP.CPP**

Đặt $F[i, x] = 1$ nếu có thể điền dấu vào i số đầu tiên và cho kết quả bằng x. Ta có công thức sau để tính F:

$F[1, a[1]] = 1$ (điều hiển nhiên)

$F[i, x] = 1$ nếu $F[i - 1, x + a[i]] = 1$ hoặc $F[i - 1, x - a[i]] = 1$.

Vì nếu $F[i - 1, x + a[i]] = 1$ thì ta thêm dấu – vào trước $a[i]$ thì sẽ được tổng bằng x; nếu $F[i - 1, x - a[i]] = 1$ thì ta thêm dấu + vào trước $a[i]$ thì cũng được tổng bằng x.

Nếu $F[n, S] = 1$ thì câu trả lời của bài toán là YES, ngược lại in ra NO.

Ta có một lưu ý khi tiến hành cài đặt bài toán này, đó là các kết quả biểu thức có thể âm nên mảng của chúng ta phải có khả năng lưu phần âm (tức là lưu được từ $-D$ đến D với D là tổng của N số). Do đó, vì mảng không có chỉ số âm nên ta chỉ cần cộng kết quả của biểu thức với 1 số lớn hơn D là sẽ thỏa mãn.

Độ phức tạp của thuật toán là $O(N*S)$, xấp xỉ $2.5*10^7$

Câu 6. Tên bài: **SPSEQ.CPP**

+ Gọi F1[i] là dãy con tăng dài nhất từ 1 đến i với mỗi $1 \le i \le n$

+ Gọi F2[i] là dãy con giảm dài nhất từ i đến n với mỗi $1 \le i \le n$. (thực chất chỉ là duyệt dãy từ n về 1, tìm dãy tăng dài nhất). Kết quả bài toán là:

$\max(2 * \min(F1[i], F2[n - i + 1]) - 1)$ với mỗi $1 \le i \le n$.

Trong kết quả ta có trừ đi 1 vì a[i] được tính 2 lần (ta tưởng tượng nó như là đỉnh của cái nón), còn F2[n – i + 1] là độ dài dãy giảm dài nhất từ i đến n.

Câu 7. Tên bài: **BEADS.CPP**

Cần chú ý đến vỏ ốc đầu tiên được cho vào chuỗi. Giả sử chuỗi ốc đầu tiên được cho vào là A[i], ta thấy dãy các vỏ được thêm vào sau cuối chuỗi sẽ tạo nên một dãy con tăng bắt đầu từ A[i], còn các vỏ được thêm vào đầu chuỗi sẽ tạo nên dãy con giảm bắt đầu từ A[i]. Do đó, với mỗi vị trí ta tìm độ dài dãy con tăng dài nhất (LIS) bắt đầu từ vị trí `i` và dãy con giảm dài nhất (LDS) bắt đầu từ vị trí i. Kết quả là `max(LIS + LDS - 1)` với mỗi vị trí i.

Ta thấy, để Bình không tốn thời gian thì 2 đoạn gỗ liên tiếp (lúc này đã được sắp xếp tăng dần theo L) phải có W tăng dần. Tức là khi đó, một số lượng đoạn gỗ thỏa mãn sẽ là 1 dãy con tăng theo chỉ số W. Lúc này, ta chỉ cần áp dụng tư tưởng bài LIS (dãy con tăng dài nhất) để áp dụng vào bài toán này theo chiều W. Bởi lẽ ta cần chia N đoạn gỗ thành ít nhất số lượng dãy con tăng này, nên mỗi dãy con tăng phải là dài nhất có thể. Lúc này, số lượng dãy con là kết quả của bài toán.

Câu 8. Tên bài: **STICK.CPP**

Đầu tiên, ta sẽ sắp xếp N đoạn gỗ này theo chiều tăng dần của độ dài L, nếu chúng có độ dài bằng nhau thì ta sẽ sắp xếp tăng dần theo trọng lượng W.

Ta thấy, để Bình không tốn thời gian thì 2 đoạn gỗ liên tiếp (lúc này đã được sắp xếp tăng dần theo L) phải có W tăng dần. Tức là khi đó, một số lượng đoạn gỗ thỏa mãn sẽ là 1 dãy con tăng theo chỉ số W. Lúc này, ta chỉ cần áp dụng tư tưởng bài LIS (dãy con tăng dài nhất) để áp dụng vào bài toán này theo chiều W. Bởi lẽ ta cần chia N đoạn gỗ thành ít nhất số lượng dãy con tăng này, nên mỗi dãy con tăng phải là dài nhất có thể. Lúc này, số lượng dãy con là kết quả của bài toán.

Câu 9. Tên bài: **NEMCHUA.CPP**

Đầu tiên ta nhận thấy chi phí 1 nhóm là chênh lệch phần tử lớn nhất và bé nhất trong nhóm, vì thế nên nếu trong nhóm đó phần tử lớn nhất giá trị là b, phần tử bé nhất giá trị là a, thì nhóm đó sẽ chứa tất cả các số từ a → b. Vì vậy ta sắp xếp dãy số $a_1, a_2 \ldots a_N$ theo thứ tự tăng dần → mỗi nhóm sẽ là 1 đoạn các số liên tiếp.

Quy hoạch động: gọi f[i][j] là chi phí nhỏ nhất chia dãy thành i nhóm khi xét từ vị trí 1 đến vị trí j → xem xét nhóm thứ i chứa bao nhiêu phần tử: f[i][j] = min(f[i-1][k] + a[j]-a[k+1]) nhóm thứ i chứa các số a[k+1], a[k+2],..., a[j]; max là a[j]; min là a[k+1] với k = 0…j-1.

Kết quả bài toán là f[m][n]. Độ phức tạp là $O(m*n^2)$

Câu 10. Tên bài: **BUNDAU.CPP**

Bài toán được phát biểu lại là cho dãy số $a_1, a_2, \ldots a_N$. Hãy tìm 1 dãy con liên tiếp có nhiều phần tử nhất có tổng các phần tử là số nguyên tố:

+ Gọi F(i) là tổng các phần tử khi xét dãy từ vị trí 1 đến vị trí i. Dễ dàng ta thấy F(0)= 0; F(i) = F(i-1) + a[i] với mọi i= 1..N. Ta duyệt tất cả các dãy con từ vị trí i đến vị trí j: ta tính tổng các phần tử chính bằng F(j) – F(i-1). Nhận thấy tổng các phần tử luôn nằm trong đoạn [0 …5.10^6], nên ta sàng nguyên tố trong đoạn này để kiểm tra tính nguyên tố của F(j) – F(i-1).

Kết quả bài toán là max của tất cả các dãy con tìm được thoả mãn đề bài.

Độ phức tạp của thuật toán là $O(n^2)$