SOLUȚII – ZIUA 1
Gaina

Vom afla pentru fiecare poziție `m<=n` costul (energia minimă) pentru a ajunge de pe poziția `m` pe poziția `n`. Costul are sens pentru valorile `m` pentru care `h_m >=h_p` , orice `p>m`.

Aşa că mai întâi aflăm aceste valori ale lui `m`. Mai exact, `hmax[a]` este cea mai mică poziție (mai mare decât `a`) pentru care se realizează maximul înălțimilor pe segmentul `[a+1,n]`. `hmax[n]=n` prin definiție.

Costul pt drumul de la pozitia `a` la pozitia `n` = cost_dist(`a, hmax[a]`) + cost_dist(`hmax[a],n`)

unde: cost_dist(`a, hmax[a]`) = costul pentru distanta de la pozitia `a` la pozitia `hmax[a]`, fără aterizări intermediare (cazul acesta este pt. economie maximă de energie), iar

cost_dist(`hmax[a], n`) se va depune în `cost[hmax[a]]`.

Costul pentru fiecare pozitie se va calcula de la dreapta la stânga, pentru că pe baza lui `cost[i]` se calculează `cost[i-1]`, iar `cost_dist[a,hmax[a]] = ( hmax[a]-a-1 )-(h[a] - h[ hmax[a] ] ), unde`

expresia (hmax[a] - a - 1) = consumul pentru zborul orizontal de la pozitia a pana deasupra pozitiei hmax[a], iar (h[a] - h[hmax[a]])= plusul de energie datorat coborarii pe verticala de deasupra pozitiei hmax[a] (de la nivelul h[a]) pana la nivelul h[hmax[a]] al pozitiei hmax[a])

Deci cost[a]= cost_dist(a, hmax[a]]) + cost[h[hmax[a]]], adica pentru a ajunge de pe pozitia a pe pozitia n, trecem prin pozitia hmax[a], iar cost_dist[a,hmax[a]] este costul pentru a ajunge de pe pozitia a pe pozitia hmax[a] fara "aterizari" intermediare. Observam ca pentru orice i cu a < i < hmax[a], avem h[i] < h[hmax[a]] <= h[a] asa ca pentru a ajunge de pe pozitia a pe pozitia hmax[a] facem un pas la dreapta si apoi coboram pana la inaltimea h[hmax[a]] si apoi mergem numai catre dreapta, iar cost_dist[a,hmax[a]] = (hmax[a]-a-1)-(h[a] - h[hmax[a]])

Rez

Vom citi codificarea circuitului într-un şir de caractere `s`.

Pentru a calcula rezistonța circuitului, vom utiliza două funcții recursive:

- funcția `rezserie()` calculează rezistonța unui circuit serie sau a unui circuit format dintr-un singur reziston
- funcția `rezparalel()` calculează rezistonța unui circuit paralel.

Funcția `rezserie()`:

Însumăm într-o variabilă (`suma`) rezistonțele circuitelor legate în serie. Acestea pot fi circuite formate:

- dintr-un singur reziston (dacă pe poziția curentă în şirul `s` se află litera `R`) ; în acest caz la variabila `suma` adunăm valoarea rezistonului.
- dintr-un circuit paralel (dacă pe poziția curentă în şirul `s` se află litera paranteză deschisă) ; în acest caz la variabila `suma` adunăm rezistonța circuitului paralel (obținută printr-un apel al funcției `rezparalel()`).

Sfârşitul circuitului serie este determinat de apariția caracterului virgulă sau paranteză închisă sau de sfârşitul şirului `s`.

Funcția `rezparalel()`:

Calculăm suma şi produsul circuitelor legate în paralel (acestea sunt circuite serie a căror rezistonță se determină prin apelul funcției `rezserie()`). Evident, sfârşitul circuitului paralel este marcat de întâlnirea caracterului paranteză închisă.

Observați că implementarea foloseşte recursivitatea indirectă.

Sortari

Soluția este $O(N^3)$ şi se bazează pe programare dinamică.
Construim vectorul `a` cu semnificația `a[i]`=numărul de şiruri de lungime `i` ce pot fi sortare folosind metoda balaurului.
Astfel `a[0] = 1`, `a[1] = 1`, `a[2] = 2`, `a[3] = 6`, `a[4] = 18`.

Să considerăm că am calculat deja `a[0]..a[i-1]`. Pentru a calcula `a[i]` fixăm poziția minimului şi a maximului (2 for-uri) şi înmulțim `a`[lungimea(`S1`)] cu `a`[lungimea(`S2`)] şi cu `a`[lungimea(`S3`)] şi adunăm produsul la `a[i]`.
Facem această înmulțire pentru că toate numerele din `S1` < toate numerele din `S2` < toate numerele din `S3`. Astfel noi ştim ce numere trebuie să conțină şirurile `S1`, `S2`, `S3`. Trebuie doar ca aceste şiruri sa fie sortabile cu metoda lui Arhirel şi de fapt ne interesează doar numărul lor şi nu şirurile efective.
Evident nu trebuie uitat ca după fiecare operație să păstrăm doar restul împărțirii la acel număr.