Problema Calafat – descriere soluţie
autor Denis-Gabriel Mită, Universitatea din București

Soluție O(N^2 + M) / O(N * M) - 20-30
Putem precalcula răspunsurile pentru fiecare subsecvență posibilă și răspunde la queryuri în O(1). Ne fixam capătul stânga și incrementăm capătul dreapta al intervalului, menținând prima și ultima apariție a fiecărei valori, actualizând răspunsul la incrementarea capătului dreapta în O(1).
Complexitatea acestei soluții este O(N^2 + M).
Analog putem calcula soluția pentru un query în O(N) cu o parcurgere a elementelor din subsecvență.
Complexitatea acestei soluții este O(N * M).

Soluție O(N + K * M * logN) - 45
Definim K ca fiind numărul de valori distincte din șir.
Astfel, la un query, pentru fiecare valoare putem căuta binar prima și ultima apariție a acesteia în cadrul subsecvenței, ceea ce este suficient pentru a calcula suma dorită. Complexitatea finală este O(N + K * M * logN).

Soluție O((N + M) * K) - 45
Definim K ca fiind numărul de valori distincte din șir.
Precalculăm în complexitate O(N * K) pentru fiecare poziție din șir ultima apariție a fiecărei valori de la stânga la dreapta și ultima apariție a fiecărei valori de la dreapta la stânga. Pentru un query, folosindu-ne de cele 2 precalculări putem afla prima și ultima apariție a fiecărei valori în cadrul subsecvenței, ceea ce este suficent pentru a calcula suma dorită.
Complexitatea finală este O((N + M) * K).

Soluție O((N + M) * sqrt(N)) - 70
Împărțim șirul în bucăți de lungime sqrt(N) și la un pas rezolvăm queryurile în complexitate O(sqrt(N)), făcând și o parcurgere a șirului in complexitate O(N).
Dacă ne fixăm o bucată [st, dr] de lungime sqrt(N), calculăm soluția pentru un query inclus in acest interval in complexitate O(sqrt(N)).
Incrementăm un indice de la dr + 1 până la N și pe parcurs calculăm soluția pentru subsecvența [dr + 1, i] și la fel ca în soluția precedentă menținem prima și ultima apariție a unei valori. Dacă găsim un query cu capătul dreapta in i și

capătul stânga în intervalul [st, dr], putem decrementa capătul stânga ce momentan este dr + 1 și să calculăm soluția până când acesta va fi identic cu capătul stânga al queryului. Practic, vom actualiza prima apariție a valorilor, iar după ce terminăm de calculat asta trebuie să avem grijă să refacem primele apariții cum erau înainte de query și să resetăm actualul capăt stânga la poziția dr + 1. Deoarece pentru fiecare query tot ce facem este să decrementăm un indice de maxim sqrt(N) ori, vom da răspunsul la query în complexitate O(sqrt(N)).
Astfel, pentru fiecare query calculăm răspunsul în O(sqrt(N)), iar pentru fiecare bucată de lungime sqrt(N) parcurgem O(N) valori.
Complexitatea finală este O((N + M) * sqrt(N)).

Soluție O(NlogN + M) - 100
O primă observație este că pentru o valoare fixată distanța maximă între prima și ultima sa apariție este suma distanțelor dintre aparițiile consecutive. Astfel, parcurgem șirul de la stânga la dreapta, iar când întâlnim o valoare actualizăm A[indicele ultimei apariții] cu diferența dintre indicele curent și indicele ultimei apariții, și apoi actualizăm indicele ultimei apariții a acestei valori. Când întâlnim un query cu capătul dreapta în indicele curent, soluția va fi dată de A[st] + A[st + 1] + ... + A[dr], adică suma pe subsecvența [st,dr] pe șirul A.
Atât un update cât și un query se pot face în complexitate O(log(N)) folosind un arbore indexat binar / arbore de intervale sau o altă structură asemănătoare.
Complexitatea finală este O(NlogN + M).