Cazan Bogdan-Marian 323CB Tema 4 AA

ID Hackerrank: cazanbogdan2000

Reducere polinomiala de k-Vertex Cover la SAT

Pentru ca transformarea noastra sa fie una valabila, vom proceda in felul urmator.
Alegem cel mult k noduri din graful nostru; dintre aceste seturi de cel mult k noduri alese, cel putin un set trebuie sa reprezinte o acoperire de valabila a grafului initial.
Astfel, un nod poate sa apara intr-o acoperire de un nod, de doua noduri, …, de k noduri. Prin urmare, vom avea nevoie de un total de n * k variabile, cate k variabile pentru fiecare nod in parte.
Asadar, variabilele noastre vor fi reprezentate prin numere, dupa regula:
-pentru nodul 1, avem variabilele 1, 2, 3, …, k;
-pentru nodul 2, avem variabilele k + 1, k + 2, k + 3, …, 2 * k
.
.
.
-pentru nodul n, avem variabilele (n-1)*k + 1, (n-1)*k + 2, (n-1)*k + 3, …, n * k.

N reprezinta numarul total de noduri, |V|.

Realizam in continuare toate posibilele seturi de cel mult k noduri:
Pentru o acoperire de un singur nod, avem variabilele 1, k + 1, 2k + 1, …, (n – 1) * k + 1 (variabilele destinate unui set de un singur nod); cum ne intereseaza cel mult un singur nod din astea, vom avea urmatoarele clauze:
(1V (k + 1) V (2 * k + 1) V … V (n-1)*k + 1) ^ (~1V~(k + 1)) ^ (~1V~(2k + 1)) ^ … ^ (~((n-2)*k + 1) V ((n-1)* + 1)).
Formula de mai sus spune ca setul poate fi format sau din nodul 1, deci variabila 1, sau nodul 2, cu variabila k + 1, sau … sau nodul n, cu variabila (n-1)*k + 1, si cel mult un nod este suficient, restul clauzelor surprinzand conditia aceasta.
Analog gandirii de mai sus, vom proceda si pentru o acoperire de 2, 3, …, k noduri, fara a mai scrie lungile formule (se deduc).

Verificam acum daca un set ales poate reprezenta o acoperire de cel mult k.
Pentru aceasta, testam apartenenta fiecarei muchii din cadrul grafului nostru la un set ales ca si posibila acoperire. O muchie din graf, care apartine de asemenea si multimii E (G = (V,E)) poate sa apartina unei acoperiri fie de un singur nod, fie de doua, …, fie de k noduri. Pe noi ce ne intereseaza este sa apartina cel putin unei acoperiri din acestea.
Astfel, pentru orice muchie (u, v) care apartine E, din graful nostru, avem o clauza de forma:
(u * k + 1) V (v * k + 1) V (u * k + 2) V (v * k + 2) V … V (u * k + k) V (v * k + k)
Intr-o astfel de clauza, vedem ca muchia formata din nodurile u si v poate sa apartina fie dintr-un set de un nod (acesta fiind implicit ori nodul u, ori nodul v, prin urmare am avea nevoie de variabilele u * k + 1 sau v * k + 1), fie dintr-un set de doua, …, fie dintr-un set de k noduri.

Desigur, avem cate o astfel de clauza pentru fiecare muchie din graf, deci un total de |E| clauze.

Formula finala se va realiza prin concatenarea (prin conjunctie) a tutror rezultatelor obtinute mai sus.

Calculam acum complexitatea transformarii (ne uitam in cod si calculam complexitatea pentru fiecare functie in parte):
-createGraph ==> O(n), unde n este numarul de noduri, |V|
-createVariables ==> O(k * n), k este acoperirea pe care vrem sa o aflam
-chooseVertices ==> O(k * n * k), length fiind de fapt n * k, dar range(i, length, k) face un pas de k, si avem n * k / k = n pasi pentru penultimul for, si pentru ultimul for la fel
-testCover ==> O(n * n * k) (se vede clar din loop-uri)
-main ==> O(n) (de la graph matrix)

Putem lua k ca fiind o variabila separata, dar pentru a arata ca transformarea apartine in P, vom lua worst case: k == n.

Vom obtine deci, prin insumare: O(n) + O(n^2) + O(n ^ 3) + O(n ^ 3) + O(n), iar complexitatea, ce e drept grosiera, dar in cel mai urat caz, va fi O(n ^ 3), care este polinomiala, oricum am privi la ea.