**Algoritmul Hopcroft-Karp**

**Complexitate**

Fie graful bipartit, cu partitiile si .

Grafului se adaugă un nod nou, notat , cu arce către toate nodurile din . Se efectuează o căutare de tip BFS din pentru primul nivel în căutarea nodurilor nesaturate, iar pentru fiecare astfel de nod întâlnit, se parcurg muchiile în manieră DFS până la întâlnirea unui alt nod nesaturat din mulţimea de noduri . Căutarea DFS nu se parcurge integral, aceasta oprindu-se după ce primul nod a fost găsit. În timpul parcurgerilor, nodurile sunt marcate pentru a nu fi vizitate din nou într-o căutare ulterioară. Complexitatea este, astfel, .

**Lema 1**:

Fie un cuplaj şi un cuplaj maxim in graful . Atunci, exista lanturi de crestere relativ la .

**Demonstratie**:

Fie . Notam cu muchiile din si cu muchiile din . Fie . De aici rezulta ca sunt cu mai multe muchii decat muchii . Deci, sunt cel putin componente care incep si se termina cu o muchie . Acestea sunt lanturi de crestere relativ la M.

**Lema 2**:

Fie un cuplaj oarecare si un cuplaj maxim a grafului G. Atunci, cel mai scurt lant de crestere relativ la M este .

**Demonstratie**:

Conform lemei 1, stim ca exista lanturi de crestere relativ la . Conform principiului lui Dirichlet (ce afirma ca daca exista obiecte dispuse in cutii, atunci o cutie contine 2 obiecte), se poate deduce ca un astfel de lant contine cel mult varfuri si deci muchii.

**Lema 3**:

Algoritmul Hopcroft-Karp necesita cel mult iteratii.

**Demonstratie**:

Dupa primele iteratii, lungimea minima a unui lant de crestere relativ la M este mai mare decat . Conform lemei 2, putem deduce ca de unde rezulta ca , deci cel mult iteratii mai sunt necesare pentru a obtine , de unde se deduce faptul ca algoritmul necesita cel mult iteratii.

Conform lemei 3, algoritmul se repeta de cel mult iar la fiecare pas parcurgerile BFS si DFS aduc o complexitate de , de unde rezulta complexitatea totala a algoritmului , sau daca neglijam gasirea varfurilor radacina care pornesc cautarea DFS.

**Corectitudine**

Fie graful bipartit, cu partitiile si .

Algoritmul, la fiecare pas, porneste din nodurile nesaturate din multimea si formeaza o multime maximala de drumuri cu noduri disjuncte de lungime minima, cu proprietatea ca ambele capete sunt noduri nesaturate, unul provenind din si celalalt din . Fiecare nod care alcatuieste drumul este marcat in momentul in care acesta este vizitat, pentru a nu putea fi vizitat ulterior. Astfel, se asigură faptul că drumurile conţin noduri disjuncte. Un astfel de drum descoperit reprezintă un lant de crestere in graful , cu muchii alternante. O muchie care face parte din cuplaj va avea orientarea inversata, arcul pornind dintr-un nod apartinand lui Y catre X. O astfel de modificare a grafului G asigură faptul că muchiile cuplate pot fi parcurse în continuare cu scopul extinderii cuplajului pentru nodurile rămase nesaturate. Mulţimea maximală de noduri disjuncte este, la fiecare pas, de lungime minimă. La pasul 1, lungimea va fi 1 deoarece avem numar arce de la X către Y. La pasul următor, vom avea lungimi impare, mai mari sau egale cu 3, datorită faptului că o muchie care face parte din cuplaj este marcată cu arc de la Y către X, ceea ce permite parcurgerea nodurilor saturate în căutarea unui nod nesaturat. Deci, trebuie ca dintr-un nod nesaturat să se parcurga un drum de noduri saturate până când se întâlneşte un alt nod nesaturat. Daca un nod este găsit, cuplajele se reformează iar cardinalitatea cuplajului de la pasul anterior creşte cu o unitate. Astfel, se asigură faptul că la fiecare pas cardinalitatea cuplajului creşte. Algoritmul se termină în momentul în care nu mai reuşeşte să satureze niciun nod, fiind deja obţinut cuplajul maxim (sau toate nodurile au fost saturate, caz în care se obţine cuplaj perfect).