Lobster Fisherman Optimization

VIJULIE DENIS-EMANUEL

Mai 2024

1 Link către Repository

Repository-ul cu codul sursă poate fi găsit la adresa: `https://github.com/DenisViju/Tema-PA.git`

2 Introducere

Scopul acestui proiect este de a dezvolta un software care să permită selectarea optimă a homarilor pentru a maximiza valoarea totală a capturii, respectând limita de capacitate a plasei.

3 Enunțul Problemei

Un pescar explorează o regiune de coastă bogată în homari, fiecare având propria sa dimensiune și valoare. Plasa pescarului are o capacitate limitată, exprimată în numărul total de centimetri pe care îi poate conține. Având o listă detaliată cu dimensiunile și valorile homarilor disponibili în acea regiune, sarcina este de a elabora o strategie prin care pescarul să selecteze homarii astfel încât să maximizeze valoarea totală a capturii, respectând în același timp limita de capacitate a plasei.

4 Algoritmi

Algoritmul folosit pentru această problemă este o variantă a problemei rucsacului (Knapsack problem) și este implementat folosind programarea dinamică.

1: **procedure** DynamicProgramming(Lista homarilor H, capacitatea plasei C)
2: $n \leftarrow$ numărul de homari din lista H
3: Initializează o matrice $V[0..n][0..C]$ pentru a stoca valorile maxime
4: **for** $i \leftarrow 0$ to n **do**
5: **for** $j \leftarrow 0$ to C **do**
6: **if** $i = 0$ sau $j = 0$ **then**
7: $V[i][j] \leftarrow 0$

```
            else if w_i ≤ j then
                V[i][j] ← max{V[i − 1][j], v_i + V[i − 1][j − w_i]}
            else
                V[i][j] ← V[i − 1][j]
            end if
        end for
    end for
    S ← lista homarilor selectați
    for i ← n down to 1 do
        if V[i][C] ≠ V[i − 1][C] then
            Adaugă homarul i la lista S
            C ← C − w_i
        end if
    end for
    return S
end procedure
```

5 Date Experimentale

Datele experimentale au fost generate aleator folosind un program C care produce seturi de date non-triviale mari și foarte mari.

6 Proiectarea Experimentală a Aplicației

Aplicația a fost proiectată pentru a fi modulară și ușor de utilizat, cu un cod sursă bine structurat și comentat.

7 Rezultate și Concluzii

8 Date Experimentale

Datele experimentale au fost generate utilizând un program C care produce seturi de date non-triviale. Aceste date au fost generate pentru a simula dimensiunile și valorile homarilor disponibili într-o regiune de coastă. Mai jos sunt prezentate câteva exemple de date generate:

- Lobster 1: Dimensiune = 2 cm, Valoare = 10 monede de aur
- Lobster 2: Dimensiune = 8 cm, Valoare = 29 monede de aur
- Lobster 3: Dimensiune = 4 cm, Valoare = 37 monede de aur
- Lobster 4: Dimensiune = 4 cm, Valoare = 20 monede de aur
- Lobster 5: Dimensiune = 8 cm, Valoare = 2 monede de aur

- Lobster 6: Dimensiune = 4 cm, Valoare = 35 monede de aur
- Lobster 7: Dimensiune = 1 cm, Valoare = 4 monede de aur
- Lobster 8: Dimensiune = 3 cm, Valoare = 30 monede de aur
- Lobster 9: Dimensiune = 2 cm, Valoare = 17 monede de aur
- Lobster 10: Dimensiune = 7 cm, Valoare = 48 monede de aur

Aceste date au fost utilizate pentru testarea algoritmului de selecție a homarilor.

Rezultatele experimentale arată că algoritmul de programare dinamică este eficient pentru optimizarea valorii capturii, maximizând valoarea totală fără a depăși limita de capacitate.