KASRLI RYUGZAK ALGORITMI

Onarkulov Maksadjon Karimberdiyevich
Farg'ona davlat univarsiteti Amaliy matematika va informatika kafedrasi dotsenti (PhD)
maxmaqsad@gmail.com
Mirzaabdullayeva Kamolatxon Abdulhafiz qizi
Farg'ona Davlat Universiteti 2-kurs talabasi
mirzaabdullayevakamolat@gmail.com

Annotatsiya. *Ushbu maqolada kasrli ryugzak muammosi va uni hal qilishda qo'llaniladigan ochko'z (greedy) algoritmning nazariy asoslari, ishlash prinsipi va amaliy yechimi yoritilgan. Muammoning matematik modeli, yechimning optimal bo'lish sharti va dasturlashdagi implementatsiyasi tahlil qilinadi. Kasrli ryugzak algoritmi logistika, moliya va resurslarni optimal taqsimlash sohalarida keng qo'llanilishi mumkinligi ko'rsatib o'tilgan.*

Kalit so'zlar*.kasrli ryugzak, ryugzak muammosi, ochko'z algoritm, kombinatorik optimallashtirish, qiymat/og'irlik nisbati, Python, optimal yechim, resurslarni taqsimlash.*

Kirish. Kombinatorik optimallashtirish muammolaridan biri bo'lgan ryugzak muammosi (Knapsack Problem) informatikada, amaliy matematikada va iqtisodiy modellashtirishda keng qo'llaniladi. Uning eng mashhur variantlaridan biri bu kasrli ryugzak algoritmi bo'lib, u klassik ryugzak muammosining soddalashtirilgan versiyasi hisoblanadi. Bu maqolada biz kasrli ryugzak algoritmi mohiyatini, ishlash prinsipini, algoritmning matematik asoslarini, amaliy misollar bilan birga tahlil qilamiz.

Kasrli ryugzak muammosi ta'rifi

Kasrli ryugzak muammosida bizga quyidagilar berilgan:

- Har biri qiymat (value) va og‘irlik (weight) ga ega bo‘lgan *n* ta element (yuk).
- Ma’lum og‘irlik chegarasiga ega bo‘lgan ryugzak.

Maqsad – ryugzakka og‘irligi **W** dan oshmaydigan tarzda narsalarni joylashtirish orqali eng yuqori qiymatni olish.

Kasrli variantda, har bir elementni butun emas, balki kasr ko‘rinishida ham ryugzakka joylashtirish mumkin. Bu uni klassik butun ryugzak muammosidan ajratib turadi.

Muammoni matematik ifodalanishi

Berilgan:

- Qiymatlar: $v = [v_1, v_2, ..., v_n]$
- Og‘irliklar: $w = [w_1, w_2, ..., w_n]$
- Ryugzak sig‘imi: W

Har bir element uchun:

$0 \le x_i \le 1$ (kasr qiymatda tanlanadi)

Maqsad:

Maksimum qilish:

$\text{maximize} \sum_{i=1}^{n} x_i * v_i$

shart bilan:

$\sum_{i=1}^{n} x_i * w_i \le W$

Greedy (ochko‘z) algoritm yondashuvi

Kasrli ryugzak muammosi ochko‘z algoritm bilan eng optimal hal qilinadigan muammolardan biridir.

Algoritm bosqichlari:

1. Har bir element uchun qiymat/og‘irlik nisbatini hisobla: v_i / w_i
2. Barcha elementlarni shu nisbat bo‘yicha kamayish tartibida sarala.
3. Ryugzak bo‘sh bo‘lsa:
   - Eng yuqori nisbatga ega elementdan boshlab, uni ryugzakka joylashtir.
   - Agar to‘liq sig‘sa, butun holatda joylashtir.
   - Aks holda, sig‘ganicha (kasr holda) joylashtir va jarayonni to‘xtat.

Asosiy afzallik:

- Yechim har doim optimal bo‘ladi.
- **Vaqt murakkabligi**: O(n log n) (saralash sababli)

Amaliy misol

Masala: Quyidagi elementlar berilgan:

Element	**Qiymat (v)**	**Og‘irlik (w)**
A	60	10
B	100	20
C	120	30

Ryugzak sig‘imi: **50**

1-bosqich: Qiymat/og‘irlik nisbati

- A: 60 / 10 = 6
- B: 100 / 20 = 5
- C: 120 / 30 = 4

Saralash: A > B > C

2-bosqich: Elementlarni joylashtirish

- A (to‘liq): og‘irlik = 10 → qolgan = 40
- B (to‘liq): og‘irlik = 20 → qolgan = 20
- C (kasrli): 20/30 = 2/3 → qiymat = 120 × 2/3 = 80

Natija:

Umumiy qiymat = 60 + 100 + 80 = **240**

C# kod: Kasrli Ryugzak Algoritmi

```
using System;
using System.Collections.Generic;

class Item
{
    public double Value { get; set; }
    public double Weight { get; set; }

    // Qiymat/og‘irlik nisbati (Greedy mezon)
    public double Ratio => Value / Weight;
}

class Program
{
    static double FractionalKnapsack(List<Item> items, double capacity)
    {
        // Qiymat/og‘irlik nisbati bo‘yicha kamayish tartibida saralash
        items.Sort((a, b) => b.Ratio.CompareTo(a.Ratio));

        double totalValue = 0;
```

```
        foreach (var item in items)
        {
            if (capacity >= item.Weight)
            {
                // Element to‘liq sig‘adi
                totalValue += item.Value;
                capacity -= item.Weight;
            }
            else
            {
                // Faqat kerakli qismini olamiz (kasrli holat)
                totalValue += item.Ratio * capacity;
                break; // Ryugzak to‘ldi
            }
        }

        return totalValue;
    }

    static void Main()
    {
        // Misol elementlar
        var items = new List<Item>
        {
            new Item { Value = 60, Weight = 10 },
            new Item { Value = 100, Weight = 20 },
            new Item { Value = 120, Weight = 30 }
        };
```

```
        double capacity = 50;
        double maxValue = FractionalKnapsack(items, capacity);

        Console.WriteLine($"Eng yuqori qiymat: {maxValue}");
    }
}
```

Tushuntirish

Item klassi:

- Har bir yuk (element) qiymat (Value) va og‘irlik (Weight) ga ega.
- Ratio — bu qiymat/og‘irlik nisbati, ya'ni ochko‘z algoritm tanlov kriteriyasi.

FractionalKnapsack() funksiyasi:

- Elementlarni Ratio bo‘yicha kamayish tartibida saralaydi.
- Ryugzak sig‘imiga qarab har bir elementni to‘liq yoki kasrli shaklda qo‘shadi.
- capacity – qancha joy qolganini ifodalaydi.
- totalValue – umumiy to‘plangan qiymatni hisoblaydi.

Main() funksiyasi:

- Test ma'lumotlari kiritiladi.
- FractionalKnapsack() chaqiriladi va natija konsolga chiqariladi.

Natija (C# dasturini ishga tushirganda):

Eng yuqori qiymat: 240

Xulosa

Kasrli ryugzak algoritmi — kombinatorik optimallashtirish muammolari ichida eng samarali va keng qo‘llaniladigan modellardan biridir. Uning asosiy ustunligi shundaki, bu algoritm greedy (ochko‘z) yondashuvi orqali har doim optimal natija beradi. Klassik (butun) ryugzak muammosidan farqli o‘laroq, kasrli ryugzak algoritmida elementlarni qisman (kasr shaklida) ryugzakka joylashtirish mumkin bo‘lgani uchun, bu muammo matematik jihatdan nisbatan soddaroq hisoblanadi.Algoritmning ishlash printsipi — har bir elementning qiymat/og‘irlik nisbatini hisoblab, ularni bu nisbat bo‘yicha kamayish tartibida tanlashdan iborat. Bu yondashuv muammoni O(n log n) vaqt murakkabligida hal qilish imkonini beradi, bu esa katta hajmdagi ma’lumotlar ustida ishlashda ham samaradorligini saqlab qoladi.

Amaliy qo‘llanilishi:

Kasrli ryugzak algoritmi quyidagi sohalarda keng qo‘llaniladi:

- **Logistika**: yuklarni yuk mashinalariga yoki konteynerlarga joylashtirishda maksimal foyda olish;
- **Moliyaviy rejalashtirish**: cheklangan budjetga eng foydali investitsiya loyihalarini tanlash;
- **Bulutli hisoblash (cloud computing)**: server resurslarini samarali taqsimlash;
- **Marketing strategiyalari**: cheklangan resurslar asosida reklama kampaniyalarini tanlash;
- **Operatsion tadqiqotlar**: harbiy, sog‘liqni saqlash, va ishlab chiqarish sohalarida optimal qaror qabul qilish.

Shuningdek, bu algoritm algoritmlarni o‘rganishda klassik misollardan biri hisoblanib, o‘quvchilar uchun greedy yondashuv asoslarini tushunishda katta yordam beradi.

Foydalanilgan adabiyotlar

1. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.** (2009). *Introduction to Algorithms*. MIT Press.

2. **Kleinberg, J., & Tardos, É.** (2006). *Algorithm Design*. Pearson Education.

3. **Sedgewick, R., & Wayne, K.** (2011). *Algorithms (4th Edition)*. Addison-Wesley.

4. **Martello, S., & Toth, P.** (1990). *Knapsack Problems: Algorithms and Computer Implementations*. Wiley.

5. **Mitzenmacher, M., & Upfal, E.** (2005). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

6. Wikipedia: Fractional Knapsack Problem

7. GeeksForGeeks: Greedy Algorithm to solve fractional knapsack