رویکرد حریصانه

روش برنامه‌نویسی پویا و مقایسه آن با رویکرد حریصانه (یادآوری و مرور)

In dynamic programming, a *recursive property* is used to divide an instance into smaller instances. In the *greedy approach*, there is no division into smaller instances.
A *greedy algorithm* arrives at a solution by making a sequence of choices, each of which simply looks the best at the moment. Each choice is *locally optimal*. The hope is that a globally optimal solution will be obtained, but this is **not always** the case.

رویه‌های اصلی در رویکرد حریصانه

1. **Selection Procedure**: chooses the next item to add to the set. The selection is performed according to a greedy criterion that satisfies some locally optimal consideration at the time.
2. **Feasibility Check:** determines if the new set is feasible by checking whether it is possible to complete this set in such a way as to give a solution to the instance.
3. **Solution Check:** determines whether the new set constitutes a solution.

قانون حداکثر: بشر همیشه سعی می کند در قبال صرف وقت ، پول ، تلاش یا احساس خود، بیشترین نتیجه را حاصل کند . در انتخاب بین کمتر یا بیشتر ، ما همیشه بیشتر را انتخاب می‌کنیم. بنابراین ، ما مردم اصولا در انجام هر کاری **حریص** هستیم . این ویژگی فی نفسه نه خوب است و نه بد . این فقط یک واقعیت است.

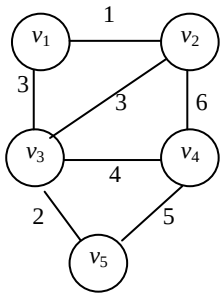
مثال: خردکردن سکه (Coin Change Problem)

درخت پوشای کمینه (Minimum Spanning Trees = MST)

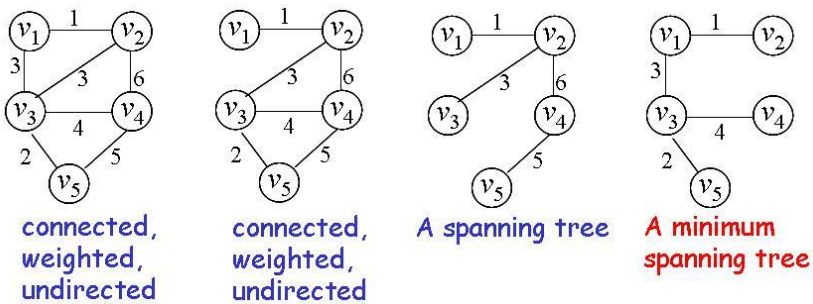
A *spanning tree* T *for* G has the same vertices V as G, but the set of edges of T is a subset F of E. We will denote a spanning tree by T= (V,F). A spanning tree T is called a *minimum spanning tree* for G if the weighted sum of F is the minimum.

$V=\{v1, v2, v3, v4, v5\}$

$E=\{(v1, v2), (v1, v3), (v2, v3), (v2, v4), (v3, v4), (v3, v5), (v4, v5)\}$



شکل۷-۱: گراف بدون جهت



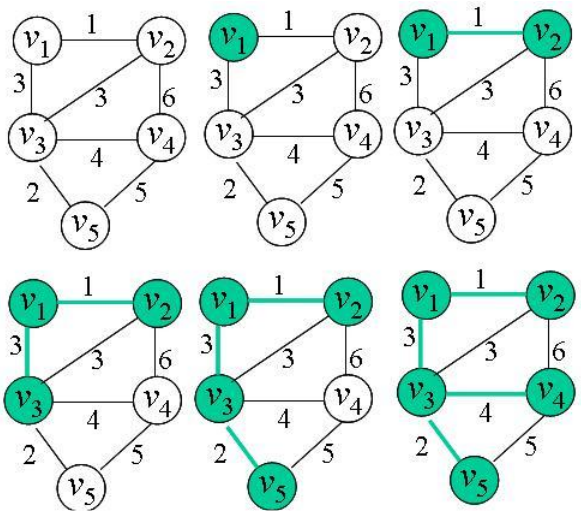
شکل۷-۲: زیرگراف‌ها و درخت‌های پوشا از گراف اصلی

پرسش: تعداد درخت‌های پوشا در یک گراف کامل K_n، چیست؟

الگوریتم پریم (Prim's Algorithm)

```
F=∅;     Y={v1};
while (the instance is not solved)
{
   select a vertex in V–Y that is nearest to Y;
    add the vertex to Y;
    add the edge to F;
    if (Y=V)        the instance is solved;
 }
```

Alg. 7-1

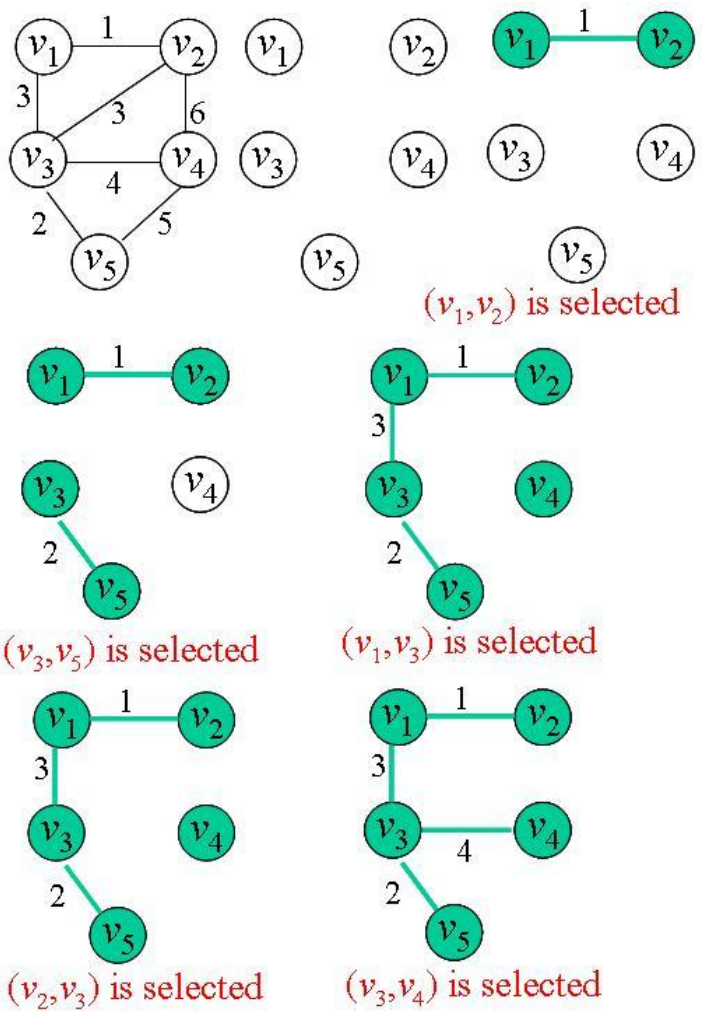


شکل ۷-۳: مراحل الگوریتم پریم

الگوریتم کروسکال (Kruskal's Algorithm)

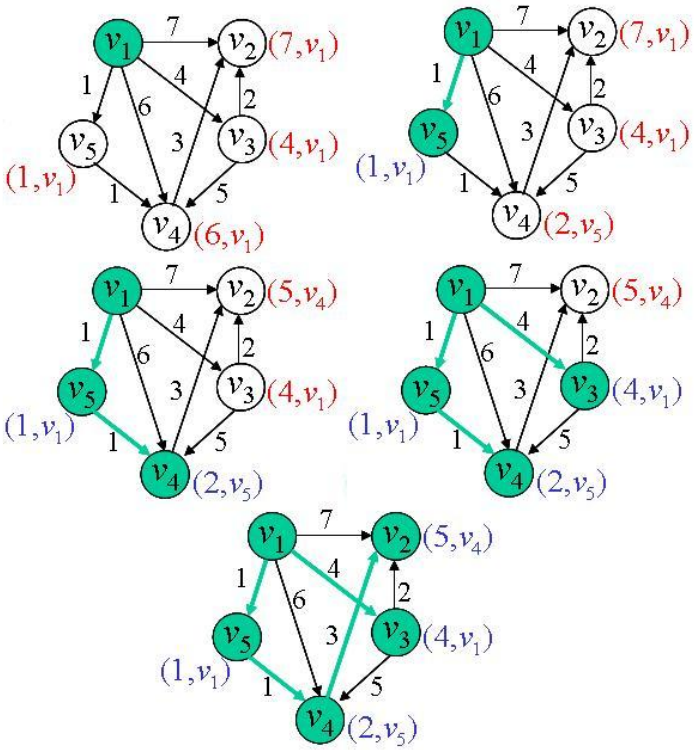
```
F=∅;
create disjoint subsets of V, one for each vertex and
containing only that vertex;
sort the edges in E in nondecreasing order;
while (the instance is not solved)
{
select next edge;
if (the edge connects two vertices in disjoint subset)
      {
        merge the subset;
        add the edge to F;
      }

  if (all the subsets are merged)
     the instance is solved;
}
```

Alg. 7-2



شکل ۷-۴: مراحل الگوریتم کروسکال

پرسش: کدام الگوریتم مناسبتر و بهتر است؟ چرا؟ (با فرض اینکه ساختمان‌داده هر دو الگوریتم، ماتریس همسایگی باشد.)



شکل۷-۵: مراحل الگوریتم دیکسترا

الگوریتم دیکسترا (Dijkstra's Algorithm)

Single-Source Shortest Paths

Y={*v*1};
F=∅;
while (the instance is not solved)
{
 select a vertex *v* from *V*–*Y*, that has a shortest path from *v*1, using only vertices in *Y* as intermediates;

add the new vertex *v* to *Y*;
add the edge (on the shortest path) that touches *v* to *F*;
 if (*Y*=*V*) the instance is solved;
}

Alg. 7-3

زمان‌بندی

زمان کل در سیستم (Time in the System)

زمان سپری شده هم برای انتظار و هم سرویس دهی(اجرا)، زمان کل در سیستم است؛

زمان کل در سیستم = زمان انتظار + زمان سرویس

مثال: زمان کل در سیستم را حساب نمایید. تفسیر زمان کل چیست؟

Task	T_1	T_2	T_3
Service Time	5	10	4

۱- زمان‌بندی ساده

پرسش: چگونه می‌توان زمان کل در سیستم را کمینه نمود؟

چند ترتیب برای مثال پیشین وجود دارد؟

Sequence	**Total Time**
[1,2,3]	5+(5+10)+(5+10+4) = 39
[1,3,2]	5+(5+4)+(5+4+10) = 33
[2,1,3]	10+(10+5)+(10+5+4) = 44
[2,3,1]	10+(10+4)+(10+4+5) = 43
[3,1,2]	4+(4+5)+(4+5+10) = 32
[3,2,1]	4+(4+10)+(4+10+5) = 37

پرسشی دیگر: روشی حریصانه برای حل مساله زمان بندی ساده پیشنهادکنید؟ مرتبه پیچیدگی زمانی آن را حساب کنید.

۲- زمان‌بندی مهلت‌دار

Task	T_1	T_2	T_3	T_4
Service Time	1	2	1	4
Deadline	1	3	2	1

۳- زمان‌بندی مهلت‌دار با بازده(با بهره)

1. In this scheduling problem, each job takes **one unit** time to finish and has a deadline and a profit.
2. If the job starts before or at its deadline, the profit is obtained.
3. The goal is to schedule the jobs so as to maximize the total profit.
4. Not all jobs have to be scheduled.
5. We need not consider any schedule that has a job scheduled after its deadline because that schedule has the same profit as one that doesn't schedule the job at all.

مثال:

Job	Deadline	Profit
1	2	30
2	1	35
3	2	25
4	1	40

چند اصطلاح و تعریف

Feasible Sequence: all the jobs in the sequence start by their deadlines. **Feasible Set:?**
Optimal Sequence: a feasible sequence with maximal total profit. **Optimal Set of Jobs?**

نکته: تفاوت مفهوم لیست(توالی) و مجموعه، در ویژگی ترتیب عناصر است.

لم

Let *S* be a set of jobs. Then *S* is feasible if and only if the sequence obtained by ordering the jobs in *S* according to nondecreasing deadlines is feasible.

الگوریتم زمان‌بندی مهلت‌دار با بازده

Problem: Determine the schedule with maximal total profit.
Inputs: number of jobs *n*, and *deadline*[*i*] for the *i*th job and sorted array of jobs in nonincreasing order.
Output: an optimal sequence *J* for the jobs.

پرسش: مرتبه زمانی الگوریتم روبرو چیست؟

```
void schedule (int  n, const int  deadline[], sequence_of_integer&  J)
{
   index  i;
    sequence_of_integer  K;
    J = [1];
    for ( i= 2; i <= n; i++)
    {
      K = J with i added according to nondecreasing values of deadline[i];
          if (K is feasible)
             J = K;
      }
}
```

Alg. 7-4

مثال: الگوریتم زمان‌بندی مهلت‌دار با بازده را روی جدول زیر، مرحله به مرحله، دنبال کنید. کارها یا وظایف در جدول زیر، بر اساس بهره (Profit) به صورت غیرافزایشی (از بزرگ به کوچک)، برای ورودی الگوریتم، مرتب‌شده‌اند.

Job	1	2	3	4	5	6	7
Deadline	3	1	1	3	1	3	2
Profit	40	35	30	25	20	15	10

کد هافمن (Huffman Code)

پرسش: اهمیت فشرده‌سازی در چیست و چه کاربردهایی دارد؟

انواع کدگذاری

۱- کدگذاری با طول ثابت

a: 00 b: 01 c: 11

۲- کدگذاری با طول متغیر

a: 10 b: 0 c: 11

نکته:

Prefix Code: no codeword for one symbol constitutes the beginning of the codeword of another symbol

مثال: در کدگذاری‌های مختلف زیر، مجموع تعداد بیت‌های لازم را محاسبه نمایید.

Character	Frequency	C1	C2	C3 (Huffman)
a	16	000	10	00
b	5	001	11110	1110
c	12	010	1110	110
d	17	011	110	01
e	10	100	11111	1111
f	25	101	0	10

پاسخ:

Bits(C1) = (16+5+12+17+10+25)3 = 255
Bits(C2) = 16(2)+5(5)+12(4)+17(3)+10(5)+25(1) = 231
Bits(C3) = 16(2)+5(4)+12(3)+17(2)+10(4)+25(2) = 212

ایده ایجاد کد هافمن

A greedy algorithm that constructs an optimal prefix code called a **Huffman code**
Assume that:

- C is a set of n characters
- Each character has a frequency f(c)
- The tree T is built in a bottom up manner

Idea:

- Start with a set of |C| leaves
- At each step, merge the two least frequent objects: the frequency of the new node = sum of two frequencies
- Use a min-priority queue Q, keyed on f to identify the two least frequent objects

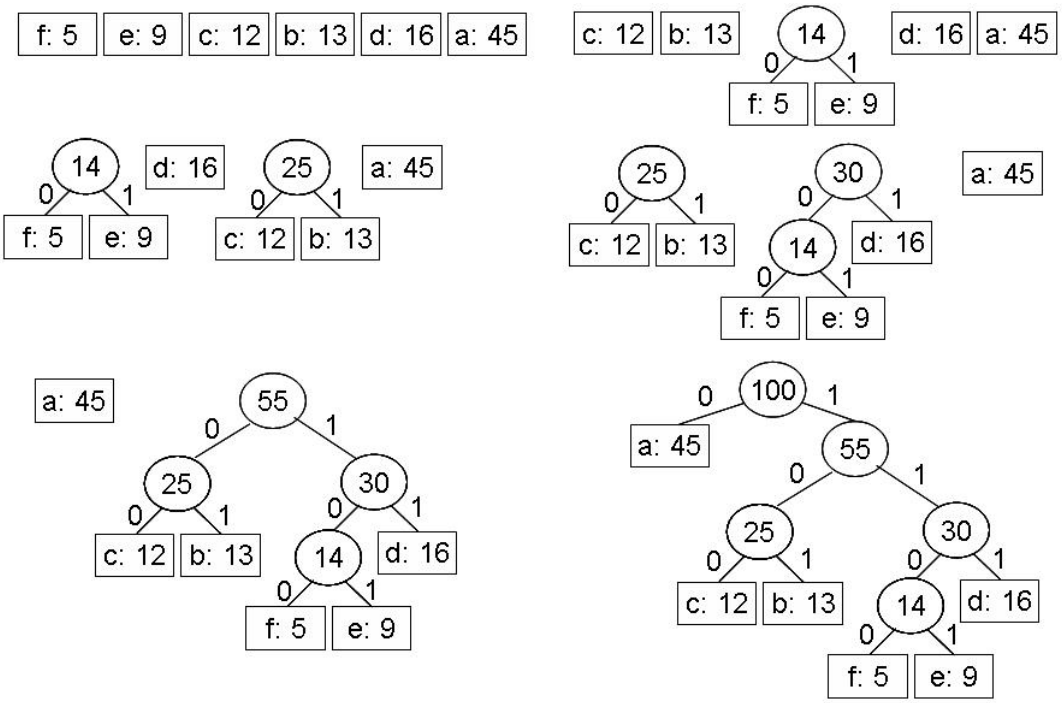
پرسش: صف اولویت‌دارچیست؟ انواع آن کدامند؟
چگونه پیاده‌سازی می‌شود؟

الگوریتم هافمن

```
HUFFMAN(C)                                  Running time: O(nlgn)
n ← |C|
Q ← C
for i ← 1 to n - 1
    do allocate a new node z
        left[z] ← x ← EXTRACT-MIN(Q)
        right[z] ← y ← EXTRACT-MIN(Q)      } O(nlgn)
        f[z] ← f[x] + f[y]
        INSERT (Q, z)
return EXTRACT-MIN(Q)
```

Alg. 7-5

مثال: الگوریتم هافمن را برای ایجاد کدهافمن با توجه به اطلاعات داده شده، مرحله به مرحله دنبال کنید.



شکل۷-۶: مراحل الگوریتم هافمن

مساله کوله‌پشتی (Knapsack Problem)

$S = \{item_1, item_2, \ldots, item_n\}$ w_i = weight of $item_i$ p_i= profit of $item_i$, W = maximum weight the knapsack can hold. Determine a subset A of S such that $\sum_{item_i \in A} p_i$ is maximized subject to $\sum_{item_i \in A} w_i \le W$

کوله‌پشتی صفر و یک (0-1 Knapsack Problem)

$[w_i]$ = [5, 10, 20], $[p_i]$ = [50, 60, 140], and W =30,

If the greedy strategy steal the largest profit per unit weight first, the profit is only 190, but the optimal is 200.

$$P[i][w] = \begin{cases} \max\{P[i-1][w],\ p_i + P[i-1][w-w_i]\}, & \text{if } w_i \le w \\ P[i-1][w], & \text{if } w_i > w \end{cases}$$

$$P[0][w] = P[i][0] = 0$$

The maximum profit is equal to $P[n][W]$.

Time Complexity: $T(n) \in O(\min(nW, 2^n))$

کوله‌پشتی کسری (Fractional Knapsack Problem)

$[w_i]$ = [5, 10, 20], $[p_i]$ = [50, 60, 140], and W =30,

The optimal profit is

50 + 140 + 60×5/10 = 220.

The Greedy Approach always yields the optimal solution for the Fractional Knapsack problem.

تمرین‌ها

تمرین۷-۱: مساله انتخاب فعالیت‌ها(Activity Selection Problem) چیست؟ بررسی کنید چگونه می‌توان آن را با رویکردحریصانه، حل کرد؟

i	p	q	r	s	t	u	v	w	x	y	z
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

تمرین۷-۲: اثبات کنید که تعداد درخت‌های پوشا در یک گراف کامل K_n، n^{n-2} است.

تمرین۷-۳: تحقیق کنید که الگوریتم هافمن در چه استانداردهای فشرده‌سازی معروفی که هم اکنون از آنها، استفاده‌می‌شود، به‌کاررفته‌است؟

تمرین۷-۴: مساله‌کوله‌پشتی صفرویک را $[w_i]$ = [5, 10, 20], $[p_i]$ = [50, 60, 140], and W =30 با روش برنامه‌نویسی‌پویا حل کنید.