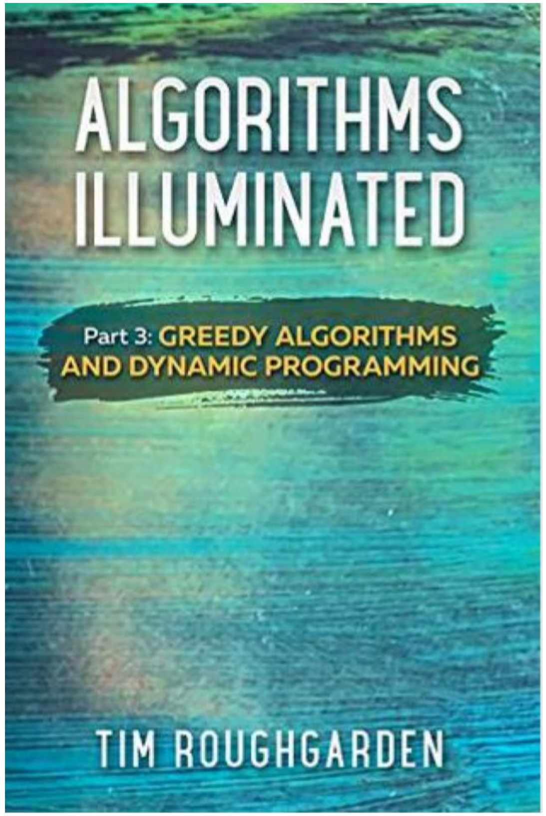
طراحی الگوریتم

(حریصانه)

دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

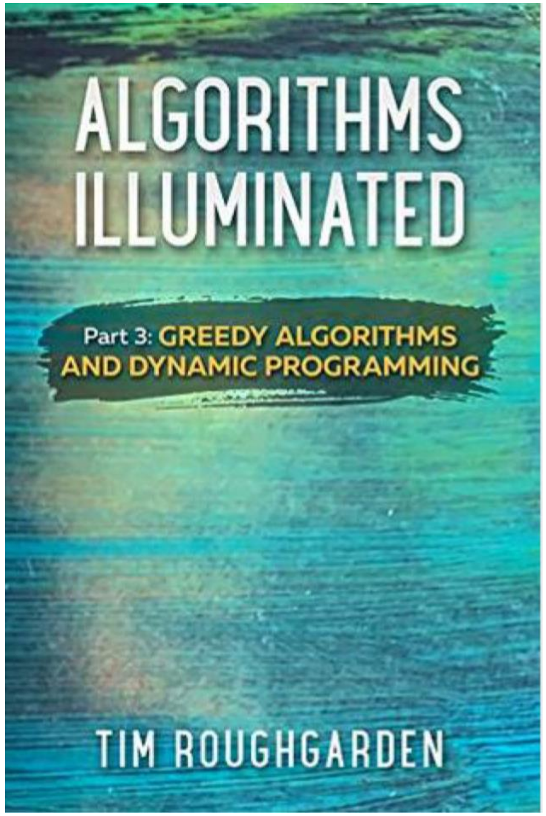
بهار ۱۴۰۰

رویکرد حریصانه



فصل پانزدهم، صفحه ۵۲

رویکرد حریصانه

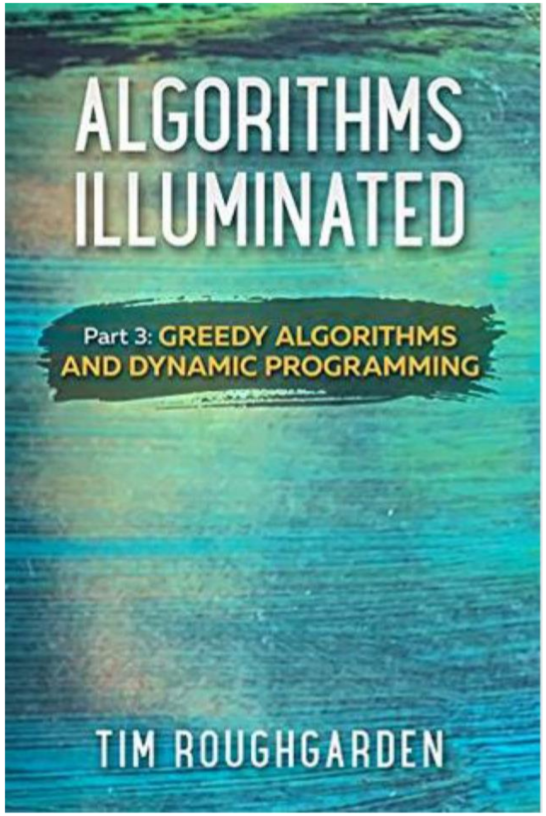


فصل پانزدهم، صفحه ۵۲

- در هر مرحله انتخابی که به نظر بهترین در لحظه می‌آید انجام می‌دهیم.
- وقتی انتخابی انجام شد، نمی‌توانیم به عقب برگردیم و آن را تغییر دهیم.
- همواره جواب درست را برنمی‌گردانند ولی برای بسیار از مسائل به درستی کار می‌کنند.

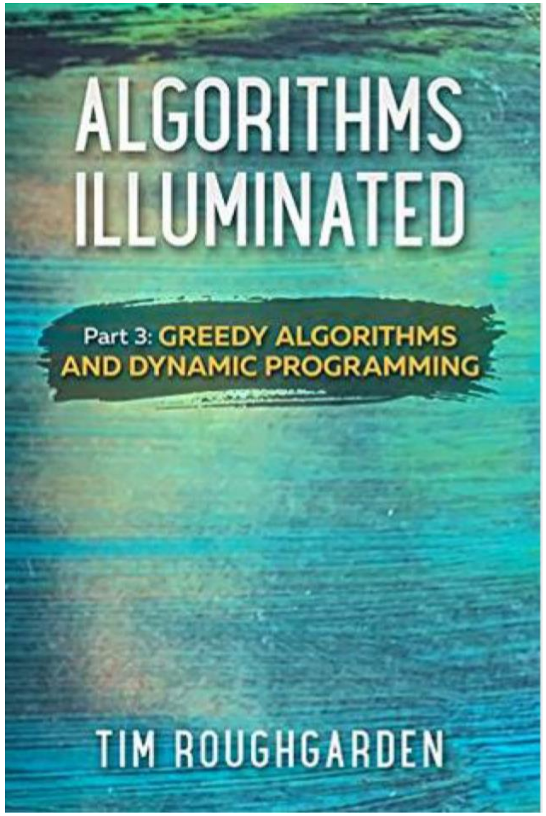
رویکرد حریصانه

- یک یا چند ایده حریصانه به سادگی به ذهن می‌رسد.
- تحلیل زمانی الگوریتم‌های حریصانه ساده است.
- اثبات درستی آن‌ها ساده نیست.



فصل پانزدهم، صفحه ۵۲

درخت پوشای کمینه



فصل پانزدهم، صفحه ۵۲

ورودی: یک گراف غیرجهت‌دار، و وزن‌های حقیقی برای هر یال.

هدف: یک درخت پوشای کمینه برای گراف ورودی که مجموع وزن یال‌های درخت کمینه باشد.

الگوریتم پریم

```
Prim
```

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

// Initialization
$X := \{s\}$ // s is an arbitrarily chosen vertex
$T := \emptyset$ // invariant: the edges in T span X
// Main loop
while there is an edge (v, w) with $v \in X, w \notin X$ **do**
    $(v^*, w^*) :=$ a minimum-cost such edge
    add vertex w^* to X
    add edge (v^*, w^*) to T
return T

درستی الگورتیم پریم

الگوریتم پریم (زمان اجرا)

Prim

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.

Output: the edges of a minimum spanning tree of G.

```
// Initialization
X := {s}     // s is an arbitrarily chosen vertex
T := ∅       // invariant: the edges in T span X
// Main loop
while there is an edge (v, w) with v ∈ X, w ∉ X do
    (v*, w*) := a minimum-cost such edge
    add vertex w* to X
    add edge (v*, w*) to T
return T
```

الگوریتم پریم (پیاده‌سازی با سریع)

Prim (Heap-Based)

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

`// Initialization`
$X := \{s\}$, $T = \emptyset$, $H :=$ empty heap
for every $v \neq s$ **do**
  if there is an edge $(s, v) \in E$ **then**
    $key(v) := c_{sv}$, $winner(v) := (s, v)$
  else   `// v has no crossing incident edges`
    $key(v) := +\infty$, $winner(v) := NULL$
  INSERT v into H
`// Main loop`
while H is non-empty **do**
  $w^* :=$ EXTRACTMIN(H)
  add w^* to X
  add $winner(w^*)$ to T
  `// update keys to maintain invariant`
  for every edge (w^*, y) with $y \in V - X$ **do**
    if $c_{w^*y} < key(y)$ **then**
      DELETE y from H
      $key(y) := c_{w^*y}$, $winner(y) := (w^*, y)$
      INSERT y into H
return T

الگوریتم کروسکال

Kruskal

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

```
// Preprocessing
T := ∅
sort edges of E by cost  // e.g., using MergeSort^26
// Main loop
for each e ∈ E, in nondecreasing order of cost do
    if T ∪ {e} is acyclic then
        T := T ∪ {e}
return T
```