به نام خدا

پیاده سازی الگوریتم حریصانه

گردآورنده: ایمان جوادی سیسی
شماره دانشجویی: ۹۸۰۱۲۳۰۶۷
درس: طراحی و تحلیل الگوریتم
استاد: سارا سادات بابایی
مقطع: کارشناسی
دانشگاه آزاد اسلامی واحد علوم و تحقیقات
۲۷ دی ۱۴۰۲

الگوریتم حریصانه یک الگوریتم تصمیم‌گیری است که در هر مرحله، بهترین تصمیم را با توجه به اطلاعات موجود در آن مرحله می‌گیرد. این الگوریتم ممکن است بهینه‌ترین راه‌حل را پیدا نکند، اما معمولاً راه‌حلی خوب و کارآمد ارائه می‌دهد.

الگوریتم حریصانه اغلب برای حل مسائلی استفاده می‌شود که دارای ساختار درختی هستند. در این مسائل، می‌توان از الگوریتم حریصانه برای یافتن کوتاه‌ترین مسیر، بیشترین ارزش یا کمترین هزینه استفاده کرد.

فرض کنید می‌خواهیم با استفاده از الگوریتم حریصانه، یک مجموعه از کارها را در کوتاه‌ترین زمان ممکن انجام دهیم. در این مسأله، هر کار دارای زمان شروع و زمان پایان است و هدف انتخاب تعداد بیشینه ای از کارها به‌گونه‌ای است که هیچ دو کاری با هم تداخل زمانی نداشته باشند. برای این کار، ابتدا باید یک تابع هدف برای محاسبه زمان انجام یک کار ایجاد کنیم.

```
def calculate_task_time(task):
  """
                    .

  Args:
    task:       .

  Returns:
          .
  """
  return task.time
```

این تابع یک کار را به عنوان ورودی دریافت می‌کند و زمان انجام آن را برمی‌گرداند. در ادامه، می‌توانیم تابع اصلی الگوریتم حریصانه را پیاده‌سازی کنیم.

```
def greedy_schedule(tasks):
    tasks.sort(key=lambda x: x[1])
    selected_tasks = []
    last_end_time = 0
```

```
    for task in tasks:
        start_time, end_time = task
        if start_time >= last_end_time:
            selected_tasks.append(task)
            last_end_time = end_time

    return selected_tasks
```

این تابع ابتدا کارها را بر اساس زمان انجام آنها مرتب می‌کند. سپس، کارها را یکی یکی به ترتیب انجام می‌دهد.

مثال: فرض کنید لیست کار های زیر را داشته باشیم

```
tasks = [ (1,4) , (3,5) , (0,6) , (5,7), (8,9)]
```

با توجه به مجموعه کتراهای فوق، مجموعه ای از کارها را با کوتاه‌ترین زمان ممکن عبارت اند از:

```
tasks = [(1, 4), (3, 5), (0, 6), (5, 7), (8, 9)]
result = greedy_schedule(tasks)
print("Selected tasks:", result)
```

```
Selected tasks: [(1, 4), (5, 7), (8, 9)]
```

به طور کلی در این مثال، تابع greedy_schedule با ورودی‌های لیست وظایف (tasks)، یک زمان‌بندی با استفاده از الگوریتم حریصانه ایجاد می‌کند. وظایف بر اساس زمان پایان به صورت صعودی مرتب می‌شوند و در هر مرحله، وظیفه‌ای که با زمان پایان کمترین تداخل با وظایف قبلی داشته باشد، انتخاب می‌شود.

لازم به ذکر است که:

۱. در این مثال، فرض کردیم که کارها به صورت مستقل قابل انجام هستند. یعنی انجام یک کار، تأثیری بر زمان انجام سایر کارها ندارد.

۲. اگر کارها به صورت وابسته قابل انجام باشند، باید الگوریتم را کمی تغییر دهیم. در این حالت، باید از یک تابع هدف متفاوت استفاده کنیم که زمان انجام کل کارها را به عنوان تابعی از ترتیب انجام کارها محاسبه کند.