دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۴

حریصانه

زهرا حسینی - ۱۳۹۸/۷/۸

جزوه جلسه ۴ام مورخ ۱۳۹۸/۷/۸ درس ساختمان‌های داده تهیه شده توسط زهرا حسینی. در جهت مستند کردن مطالب درس ساختمان های داده، بر آن شدیم که از دانشجویان جهت مکتوب کردن مطالب کمک بگیریم. هر دانشجو می‌تواند برای مکتوب کردن یک جلسه داوطلب شده و با توجه به کیفیت جزوه از لحاظ کامل بودن مطالب، کیفیت نوشتار و استفاده از اشکال و منابع کمک آموزشی، حداکثر یک نمره مثبت از بیست نمره دریافت کند. خواهش‌مند است نام و نام خانوادگی خود، عنوان درس، شماره و تاریخ جلسه در ابتدای این فایل را با دقت پر کنید.

۱.۴ روش های طراحی الگوریتم ها [۱]

در ادامه ی بخش اول،آشنایی کلی با الگوریتم،هدف این است که مسئله داده شده را با بهترین راه ،حل کنیم. سه روش کلی که به ترتیب زیر به بررسی هریک میپردازیم:

- الگوریتم حریصانه
- الگوریتم تقسیم وحل
- الگوریتم برنامه نویسی پویا

۱.۱.۴ [۲] حریصانه

در این روش مسئله قدم به قدم کوچک میشود.روش کوچک شدن با یک انتخاب صورت میگیرد،باید توجه کرد که جواب بهینه حذف نشود.

۲.۱.۴ تقسیم و حل

مسئله را به دو یا چند قسمت تقسیم میکنیم،هر زیر مسئله را مستقل از دیگری حل میکنیم در نهایت با توجه به صورت سوال جواب های به دست آمده را باهم مقایسه ، جمع و ... میکنیم.باید دقت کرد که زیرمسئله ها

هم پوشانی نداشته باشند.این روش معمولا به صورت بازگشتی انجام میشود .

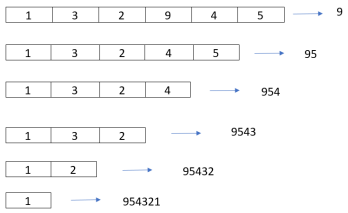
۳.۱.۴ برنامه نویسی پویا

روش همچون روش تقسیم و حل بر پایه‌ی تقسیم مسئله بر زیرمسئله‌ها کار می‌کند. داده‌های زیرمسئله وابسته به هم بوده و یا با هم اشتراک دارند یا به عبارتی هم‌پوشانی دارند. در این روش هر زیر مسئله یکبار حل میشود یعنی حافظه ایی داریم که پاسخ های به دست آمده را ذخیره میکند و دیگر زیرمسئله ایی دوباره حل نمی شود.

۲.۴ مثال های الگوریتم حریصانه

۱.۲.۴ حداکثر حقوق

ورودی آرایه ایی از اعداد است ۱،۳،۲،۹،۴،۵ با این اعداد بیشترین حقوق پیشنهادی خود را تولید کنید.
به روش حریصانه بزرگترین عدد موجود را پیدا میکنیم و آن را درچپ ترین جایگاه قرار میدهیم ،بزرگترین عدد ۹ است آن را انتخاب میکنیم حال یک انتخاب صورت گرفته و مسئله کوچک میشود در واقع عدد انتخاب شده را در هر مرحله از آرایه ورودی حذف میکنیم.



شکل ۱.۴: حداکثر حقوق

اثبات اینکه جواب بدست آمده بهترین جواب است از برهان خلف استفاده میکنیم به این صورت که عدد ۹ اگردر جایگاهی به غیر از چپ ترین جایگاه قرار بگیرد با جا به جا کردن عدد ۹ مقدار بیشتری به دست می

آید پس فرض خلف باطل است.

```
Input: Digits
Output: answer
Function MaxSalary(Digits):
    answer ← emptystring
    while Digits is not empty do
        maxDigit ← −∞
        foreach digit ∈ Digits do
            if digit ≥ maxDigit then
                maxDigit ←digit
            end
            maxDigit append to answer
            remove maxDigit from Digits
        end
        return answer
End Function
```

Algorithm 1: Maximizing Your Salary

۲.۲.۴ حداقل انتظار بیماران

افرادی در مطب پزشک منتظر هستند،هر کدام از این افراد مدت زمان متفاوتی با پزشک ملاقات میکند،چگونه میزان منتظر بودن هریک از افراد کمترین مقدار ممکن میشود؟ نفر اول انتظاری نمیکشد،نفر دوم به اندازه مدت زمانی که نفر اول نزد پزشک است منتظر میماند ،نفر سوم به اندازه مجموع این زمان برای نفر اول و دوم و به همین ترتیب نفر آخر به اندازه مجموع زمان های افراد به جز زمان خودش منتظر خواهد ماند.

$$t_1 * (n-1) + t_2 * (n-2) + ... + t_{n-1} * (1)$$

همانطور که مشخص است ضریب نفر اول(تعدادی که زمان نفر اول محاسبه میشود) ،بزرگترین ضریب است پس باید نفر اول با کمترین زمان انتخاب شود و انتخاب ها به صورت صعودی باشند یعنی نفر آخر بیشترین زمان را داشته باشد.

برای اثبات درستی راه حلمان از برهان خلف استفاده میکنیم در این حالت

$$t_i$$

ایی را فرض میکنیم و آن را با

$$t_{i+1}$$

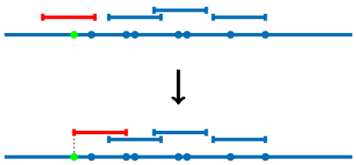
جا به جا میکنیم، اگر فرض ابتدایی ما مبنی بر بهینه نبودن جوابمان درست باشد باید

$$t_{i+1} - t_i$$

منفی شود ولی میدانیم که زمان ها را به ترتیب صعودی مرتب کرده ایم پس این اختلاف مثبت بوده و فرض خلف باطل است.

۳.۲.۴ جشن تولد

افراد حاضر در یک جشن تولد را میخواهیم به گونه ایی دسته بندی کنیم که اختلاف سنی هر یک از افراد گروه با سایر اعضا حداکثر دو سال باشد.همچنین قصد داریم برای هر گروه یک مربی استخدام کنیم، هدف در این مسئله پیدا کردن کمترین تعداد مربی ممکن است. ابتدا افراد را از کوچکترین سن به بزرگترین سن مرتب میکنیم،سپس اولین نفر را انتخاب میکنیم و تا جایی که اختلاف سنی اولین نفر و آخرین نفر حداکثر دو سال باشد ، انتخاب کردن را ادامه میدهیم.حال نفر بعدی که انتخاب میشود اولین نفر گروه بعد و مانند مرحله قبل ادامه میدهیم.



شکل ۲.۴: جشن تولد

فرض کنید

$$x_1 \leq x_2 ... \leq x_n$$

Input: points=

$$x_1, ..x_n$$

Output: *Segments*
Function *PointsCoverdSorted(points)*:
    Segments ←emptylist
    $left \leftarrow 1$
    while $left \leq n$ **do**
        $(l, r) \leftarrow (xleft, xleft + 2)$
        (l, r) *append to Segments*
        $left \leftarrow left + 1$
    end
    return *Segments*
End Function

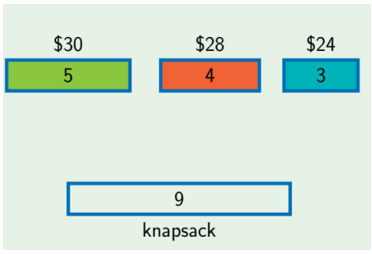
Algorithm 2: Birthday Party

انتخاب بهینه انتخابی است که چپ ترین نقطه را پوشش دهد و بازه ایی باشد که با آن نقطه آغاز شود.

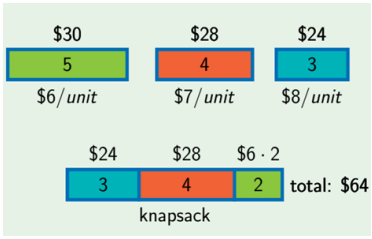
۴.۲.۴ کوله پشتی

فرض کنید میخواهیم از یک فروشگاه دزدی کنیم(!)،کوله پشتی ایی با ظرفیت مشخص داریم،انتخاب های ما به چه صورتی باشد که بیشترین درآمد حاصل شود؟ با توجه به اینکه اشیا دارای حجم متفاوتی هستند،باید اشیایی را انتخاب کنیم که نسبت به حجمشان ارزش بیشتری داشته باشند.برای اشیا خاصیت چگالی را به این صورت که میزان ارزش تقسیم بر حجم شی تعریف میکنیم حال اشیا را بر اساس چگالی اشان به صورت نزولی مرتب میکنیم و انتخاب را تا جایی که ظرفیت کوله پشتی اجازه میدهد ادامه میدهیم. فرض کنید طبق شکل زیر کوله پشتی ایی با ظرفیت ۹ و اشیایی با اطلاعات زیر در اختیار داریم

حال برای هریک از اشیا چگالی تعریف میکنیم،ابتدا شی که بیشترین چگالی را دارد اگر حجمش از ظرفیت داده شده تجاوز نکند به طور کامل انتخاب میشود و بقیه انتخاب ها هم مانند مرحله ی قبل برای حجم باقی مانده صورت میگیرد.باید به این نکته توجه داشت که میتوان برای تکمیل کردن ظرفیت کوله پشتی بخشی از شی را انتخاب کرد.



شکل ۳.۴: کوله پشتی۱



شکل ۴.۴: کوله پشتی۲

Input: W,weights= $w_1, ..w_n$,values= $v_1, ..v_n$

Output: *Segments*

Function *KnapsackFast*$(W, weights, values)$**:**

```
amounts ←[0,0,..,0]
totalValue ← 0
for i ← 1 to n do
    if W is 0 then
        return (totalValue, amounts)
    end
    a ←min(w_i, W)  totalValue← totalValue + a ∗ (v_i ÷ w_i)
    w_i ←(w_a)  amounts_i ←(amounts_i + a)  W← W − a
end
return (totalValue, amounts)
```

End Function

Algorithm 3: Knapsack

Bibliography

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[2] K. Schwarz, "https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/13/small13.pdf." `https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/13/Small13.pdf/`. Accessed: 2013-07-24.