به نام یگانه معبود بخشنده مهربان

طراحی الگوریتم ها

Design and Analysis of Algorithms

گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه اصفهان

ترم دوم سال تحصیلی ۹۰–۹۱

ارائه دهنده : پیمان ادیبی

روش عقبگرد

Backtracking

مفاهیم

- در مسایلی که هدف یافتن یک دنباله از انتخابها برای رسیدن به یک پاسخ است، در هر مرحله انتخابهای مختلفی وجود دارد.
- اگر در هر مرحله تصمیم گیری بتوان تشخیص داد که برخی از انتخابهای موجود ما را به جواب نمیرساند، دیگر آنها را انتخاب نمیکنیم، و در زمان صرفه جویی میشود. روش عقبگرد چنین رویکردی را دنبال میکند.
- عقبگرد یک نسخه بهبود یافته از جستجوی عمق اول در گراف یا درخت فضای حالت مسأله میباشد، که با هرس کردن گراف یا درخت جستجو، میتواند زمان رسیدن به پاسخ را کاهش دهد.
- الگوریتمهای عقبگرد کاهش زمان را برای همه نمونه مسأله ها تضمین نمیکنند، بلکه برای نمونه های بزرگ اغلب تسریع ایجاد میکنند.
- مثال: ...

شکل کلی عقبگرد

- گره غیر امیدبخش (nonpromising node) : گره ای از درخت فضای حالت که قطعاً میدانیم ما را به پاسخ (یا پاسخ بهتر) نمیرساند.
- گره امید بخش (promising node): گره ای که غیر امید بخش نباشد.
- بررسی امید بخش بودن نباید آنقدر زمانبر باشد که تسریع حاصل از هرس درخت را کم اثر نماید.
- دو شکل کلی برای الگوریتمهای عقبگرد:

قابل فهم تر

```
void checknode (node v)
  {
  node u;

  if (promising(v))
     if (there is a solution at v)
        write the solution;
     else
       for (each child u of v)
          checknode(u);
  }
```

کارآمدتر

```
void expand (node v)
  {
  node u;

  for (each child u of v)
       if (promising(u))
          if (there is a solution at u)
             write the solution;
          else
                expand (u);
  }
```

مثال: مسأله n وزیر

- میخواهیم تعداد n مهره وزیر را در خانه های یک صفحه شطرنج $n \times n$ بنحوی قرار دهیم که هیچ دو وزیری یکدیگر را تهدید نکنند (یعنی در یک سطر یا ستون یا قطر قرار نگیرند).
- برای کاهش تعداد حالات فرض میکنیم در هر سطر دقیقاً یک وزیر قرار میگیرد.
- تعداد گره های درخت فضای حالت :

$$1+n+n^2+\ldots+n^n=(n^{n+1}-1)/(n-1)$$

- الگوریتم عقبگرد: الگوریتم 5.1 کتاب. مثال: n=4 ...

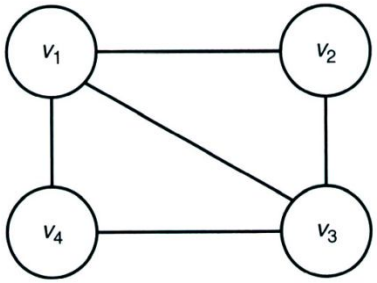
این الگوریتم تمام پاسخها را ایجاد میکند. اگر تنها یک پاسخ نیاز بود باید پس از اجرای دستور cout از برنامه خارج شویم (با exit).

- تحلیل نظری زمان اجرای الگوریتمهای عقبگرد معمولاً دشوار است. بجای آن به دو طریق میتوان عمل نمود:
  - ارزیابی عملی: مقایسه تعداد گره های بررسی شده با عقبگرد و بدون عقبگرد
  - تخمین کارایی با روشهای آماری (مانند روش مونت کارلو)

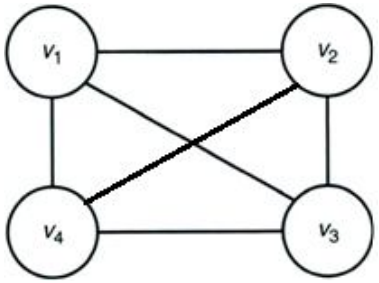
ارزیابی عملی برای مثال فوق: ...

مثال: مسأله رنگ آمیزی گراف

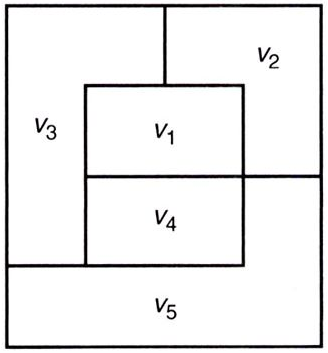
- **مسأله رنگ آمیزی m (m-coloring):** یافتن تمام راه حلها برای تخصیص رنگ به رأسهای یک گراف بدون جهت، بنحوی که هیچ دو رأس مجاوری همرنگ نباشند، با داشتن حداکثر m رنگ متفاوت.
- مثال: برای m=2 گراف مقابل جواب ... :

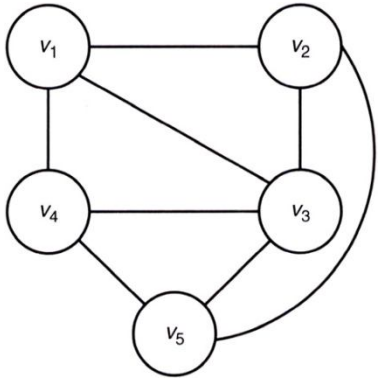


- **گراف مسطح (planar):** گرافی که بتوان آن را روی صفحه بگونه ای رسم کرد که هیچ دو یالی یکدیگر را قطع نکنند.
- مثال: گراف روبرو مسطح ... :



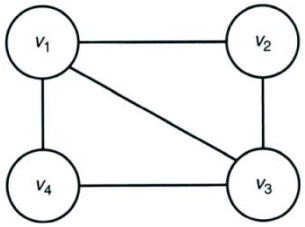
- **کاربرد در رنگ آمیزی نقشه ها:** متناظر با هر نقشه یک گراف مسطح وجود دارد.
- مثال:





مسأله رنگ آمیزی گراف

- الگوریتم:
- درخت فضای حالت: گره ها در هر سطح بیانگر رنگهای منسوب به یکی از رأسهای گراف است.
- تذکر: کلمه "رأس" برای گراف و "گره" برای درخت فضای حالت بکار میرود.
- مثال: ...




```
void m_coloring (index i)
   {
  int color;
  if (promising (i))
        if (i == n)
           cout << vcolor [1] through vcolor [n];
        else
           for (color = 1; color <= m; color++){
                 vcolor [i + 1] = color;
                 m_coloring (i + 1);
              }
   }

bool promising (index i)
   {
  index j;
  bool switch;

  switch = true;
  j = 1;
  while (j<i && switch){
     if (W[i][j] && vcolor[i] == vcolor[j])
        switch = false;
     j++;
     }
return switch;
     }
```

فراخوانی سطح بالا:
m_coloring(0)

- تعداد گره های درخت فضای حالت:

$$1 + m + m^2 + \cdots + m^n = \frac{m^{n+1} - 1}{m - 1}$$

- مسأله **NP-Complete** است.
در بدترین حالت نمایی است، اما برای مثالهای بزرگ کارایی میتواند بهتر باشد.

روش عقبگرد در مسایل بهینه سازی

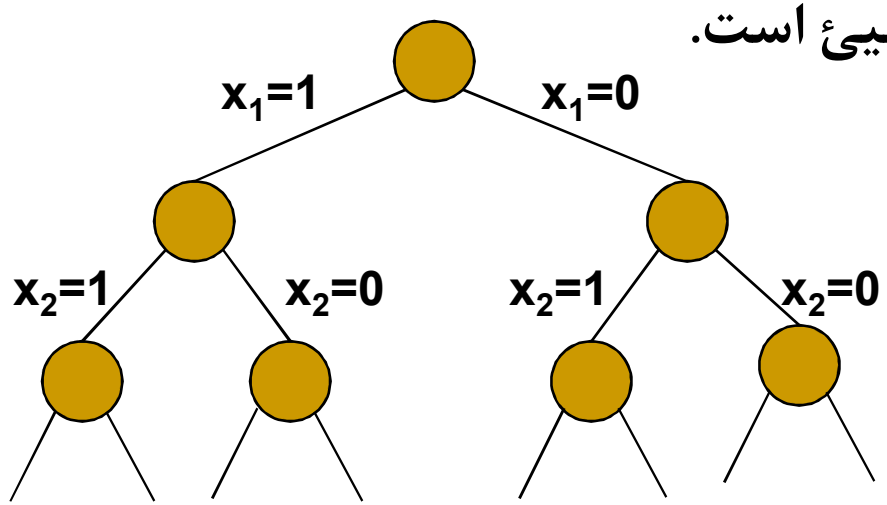
- در مسایل قبلی هرگاه به یک گره برگ میرسیدیم که قابل قبول بود، یک پاسخ شدنی پیدا میشد و میتوانستیم کار را خاتمه دهیم.
- اما در مسایل بهینه سازی پیش از بررسی تمام گره های قابل قبول درخت، پاسخ بهینه معلوم نمیشود.
- شکل کلی عقبگرد در مسایل بهینه سازی:

```
void checknode (node v)
   {
  node u;

  if (value(v) is better than best)
        best = value(v);
  if (promising(v))
        for (each child u of v)
              checknode(u);
   }
```

مثال: مسأله کوله پشتی صفر و یک

- M ظرفیت کوله پشتی، I_1 تا I_n اشیاء با ارزشهای p_1 تا p_n و وزنهای w_1 تا w_n ، انتخاب شیئ i با $x_i=1$ مشخص میشود
- درخت فضای حالت:
  - شاخه سمت چپ به معنای انتخاب یک شیئ و شاخه سمت راست به معنای عدم انتخاب شیئ است.



- profit = ارزش اشیاء انتخاب شده تا مرحله جاری
- weight = وزن اشیاء انتخاب شده تا مرحله جاری

بررسی امیدبخش بودن گره ها

1- اگر $weight \geq M$ باشد، گره غیر امیدبخش است (و لذا بسط داده نمیشود).

2- برای گره مرحله جاری یک حد بالای سود قابل حصول از آن گره به نام **bound** بدست می‌آوریم. بیشترین سودی که در گره های بسط یافته تاکنون (با برقراری شرط ظرفیت) حاصل شده را **maxprofit** مینامیم. اگر $bound \leq maxprofit$ باشد، گره غیر امیدبخش است (و لذا بسط داده نمیشود).

- **bound** = سود حاصل اگر از مرحله جاری به بعد مسأله را بصورت نسخه کسری فرض کرده و اشیاء را بطور حریصانه، به ترتیب غیر صعودی p_i/w_i ها انتخاب کنیم.
- توجه: در عمل انتخاب اشیاء بصورت حریصانه صورت نمیگیرد، بلکه این کار تنها برای یافتن حد بالایی **bound** و بصورت فرضی انجام میشود.

بررسی امیدبخش بودن گره ها

- فرض میکنیم گره جاری در سطح i درخت است. همچنین فرض میکنیم در حین انتخابهای حریصانه فرضی مذکور، با انتخاب I_k است که وزن کل اشیاء بیش از ظرفیت کوله پشتی میشود. حال تعریف میکنیم:
- **totweight** = وزن کل قبل از برداشتن I_k در انتخابهای حریصانه

$$totweight = weight + \sum_{j=i+1}^{k-1} w_j$$

- حال حد بالایی مذکور بشکل زیر محاسبه میشود:

$$bound = \underbrace{profit}_{\text{سود حاصل تا این مرحله از الگوریتم عقبگرد}} + \underbrace{\sum_{j=i+1}^{k-1} p_j}_{\text{سود حاصل از انتخابهای حریصانه فرضی بصورت صفر و یک}} + \underbrace{\frac{M - totweight}{w_k} \times p_k}_{\text{سود حاصل از کسری از } I_k \text{ که ظرفیت کوله پشتی را تکمیل میکند}}$$

مسأله کوله پشتی صفر و یک با روش عقبگرد

- تعداد گره های بررسی شونده در بدترین حالت از مرتبه نمایی است که برای حالت زیر رخ میدهد:

$$M=n,\quad p_1=\ldots=p_{n-1}=1,\quad p_n=n$$
$$w_1=\ldots=w_{n-1}=1,\quad w_n=n \quad \rightarrow \quad \Theta(2^n)$$

- مقایسه:
  - برنامه ریزی پویا: $\Theta(\min(2^n, nM))$
  - عقبگرد: $\Theta(2^n)$
- در عمل مشاهده شده که عقبگرد سریعتر است
- کاهش زمان بررسی امید بخش بودن:
  - تا زمانیکه در عمق درخت با شاخه های سمت چپ پیش میرویم، مقدار bound تا رسیدن به k سطح از شروع این حرکت بدون تغییر میماند، و نیاز به محاسبه مجدد ندارد.

مسأله کوله پشتی صفر و یک با روش عقبگرد

- مثال عددی: ...

√524