

روش‌های جستجوی ناآگاهانه

(Uninformed Search Methods)

1. جستجوی عمقی (Depth-First Search - DFS)
2. جستجوی سطحی (Breadth-First Search - BFS)
3. جستجوی عمقی افزایشی (ترکیب DFS و BFS) (Iterative Deepening Search - IDS)
4. جستجوی هزینه یکنواخت (Uniform-Cost Search - UCS)

ارزیابی جستجوی عمقی (DFS)

1. کامل بودن (Complete):

- اگر m (عمق بیشینه) نامحدود باشد، ممکن است DFS در یک مسیر بی نهایت گیر کند و هرگز راه حل را پیدا نکند.
- فقط در صورتی که از حلقه ها جلوگیری کنیم، کامل خواهد بود.

2. بهینه بودن (Optimal):

- خیر، DFS اولین راه حل موجود در سمت چپ درخت را پیدا می کند، بدون توجه به عمق یا هزینه آن.

3. پیچیدگی زمانی (Time complexity):

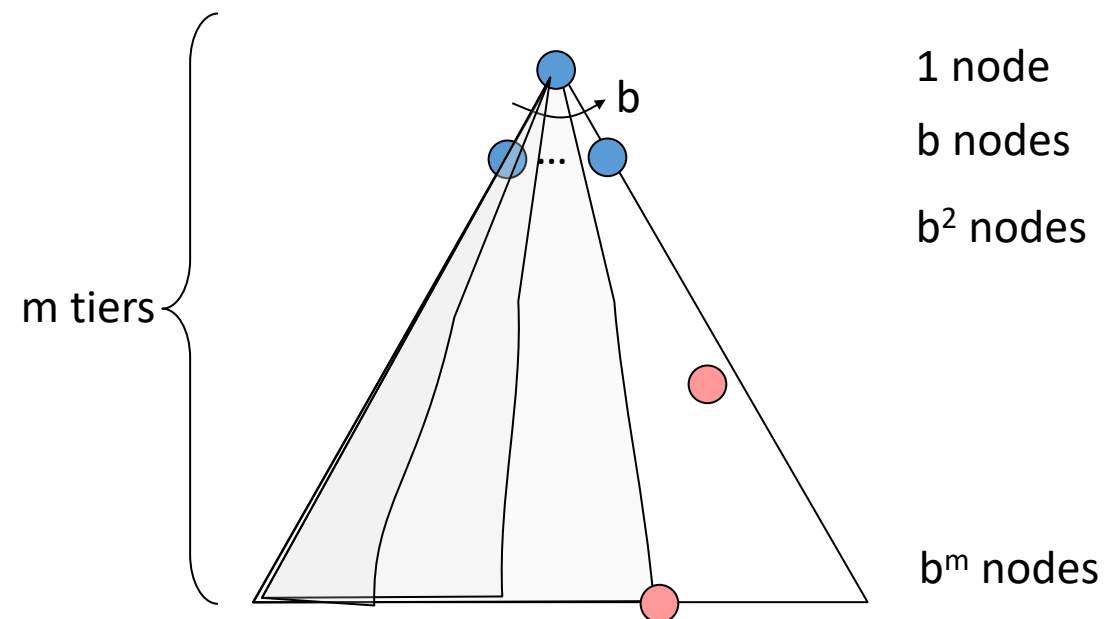
- اگر m محدود باشد، زمان اجرای آن برابر است با: $O(b^m)$
- در بدترین حالت، DFS تمام گره های یک مسیر را تا عمق m بررسی می کند.

$$1 + b + b^2 + b^3 + \dots + b^m$$

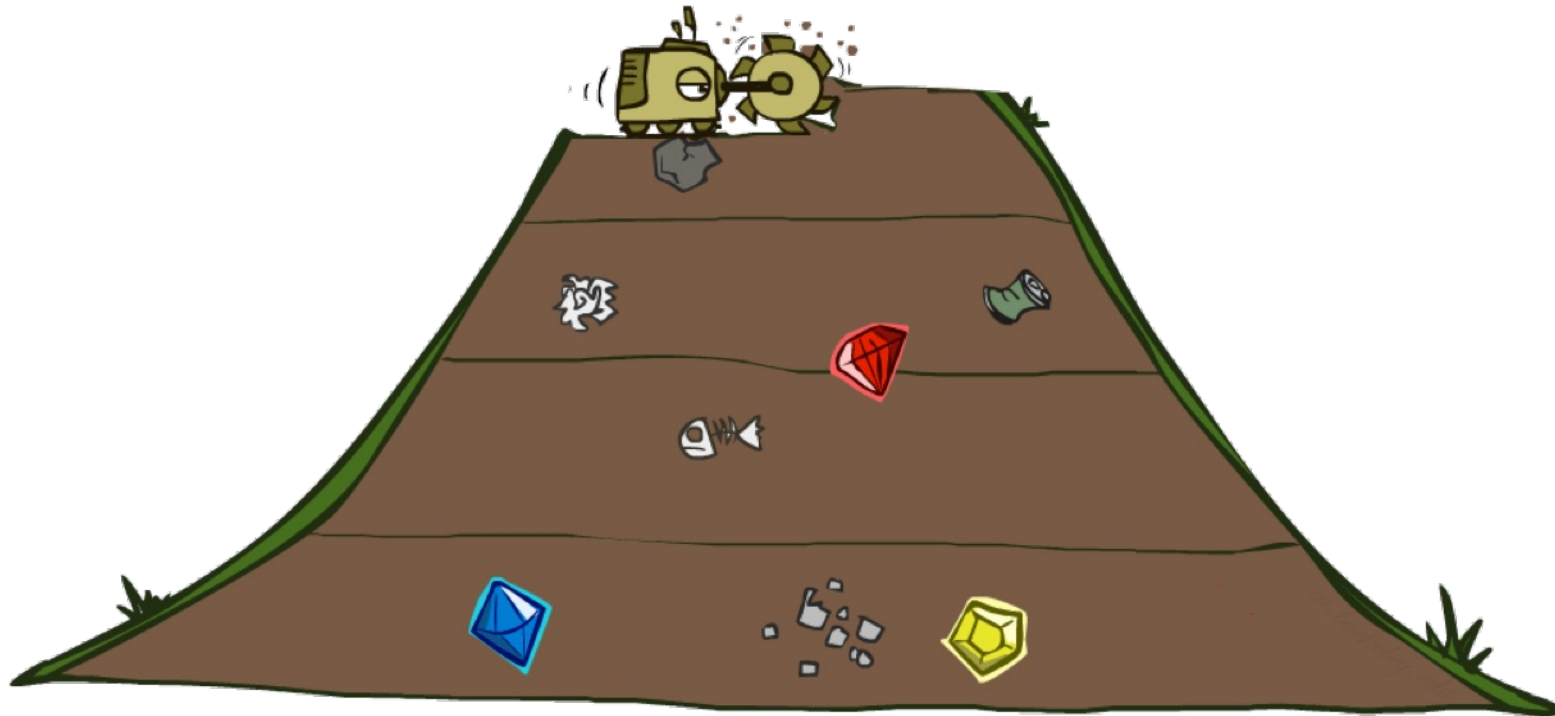
4. پیچیدگی حافظه (Space complexity):

- فضای مورد نیاز برابر است با $O(bm)$.
- فقط گره های مسیر جاری از ریشه تا گره فعلی در حافظه ذخیره می شوند.
- در عمق m ، مسیر دارای حداکثر m گره است، و در هر سطح حداکثر b گره sibling (برادر) در حافظه باقی می ماند.

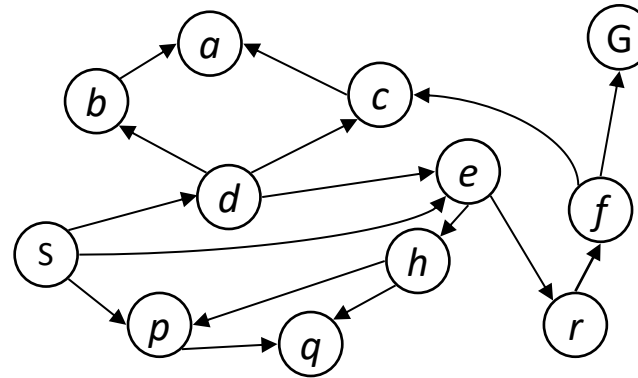
$$O(bm)$$



۲. جستجوی سطحی (Breadth-First Search - BFS)



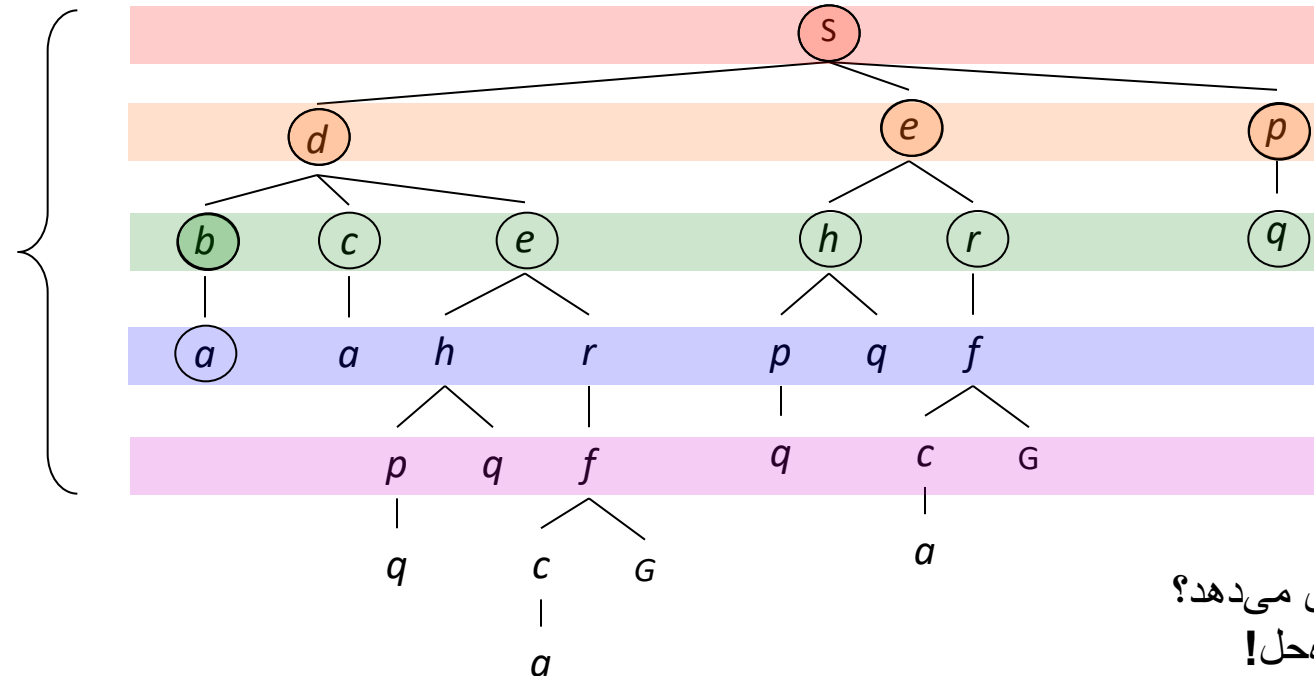
۲. جستجوی سطحی



استراتژی: ابتدا کم عمق ترین گره را گسترش دهید.

پیاده سازی: حاشیه (Fringe) یک صف FIFO است.

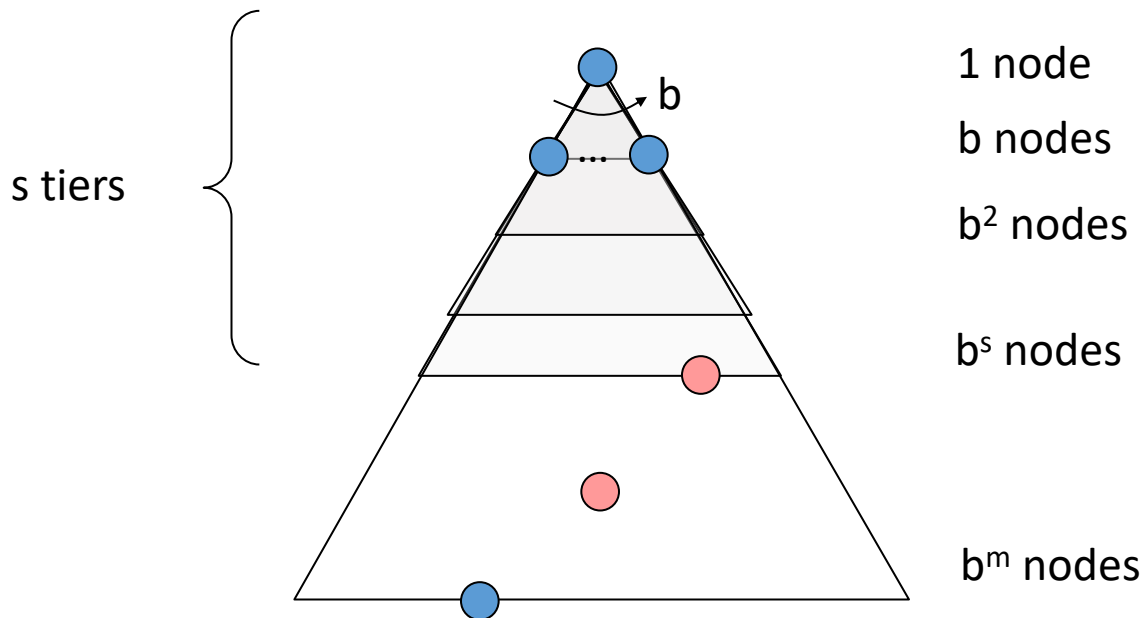
سطوح
جستجو



سوال: جستجوی سطحی کدام گره ها را گسترش می دهد؟
✓ تمام گره های بالاتر از کم عمق ترین گره راه حل!

ارزیابی جستجوی سطحی (BFS)

اگر عمق کم عمق ترین راه حل را s در نظر بگیریم:



1. کامل بودن (Complete):

▪ اگر s متناهی باشد و راه حلی وجود داشته باشد، بله!

2. بهینه بودن (Optimal):

▪ در صورتی که هزینه ها همگی 1 باشند

3. پیچیدگی زمانی (Time complexity):

▪ در بدترین حالت، BFS تمام گره های تا عمق s را بررسی می کند، که

تعدادشان برابر است با: $1 + b + b^2 + \dots + b^s \approx O(b^s)$

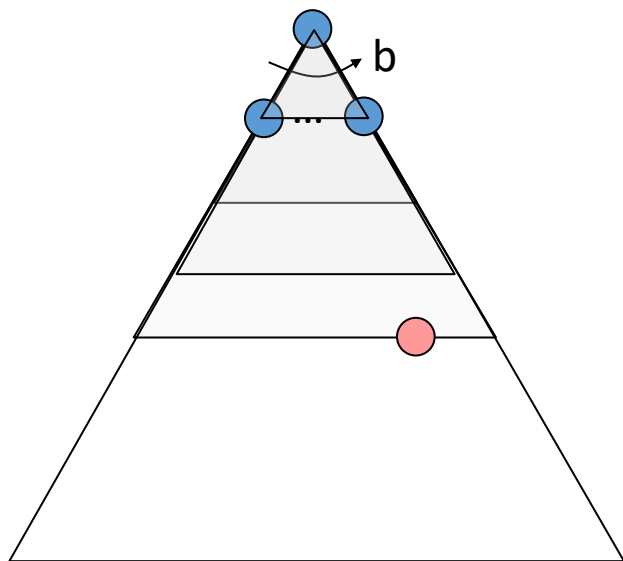
4. پیچیدگی حافظه (Space complexity):

▪ در بدترین حالت، آخرین سطح گره ها (یعنی تمام گره های عمق s) در حافظه قرار دارند، که تعدادشان تقریباً b^s است.

▪ BFS از حافظه ی نمایی استفاده می کند که معمولاً بزرگ ترین مشکل آن است.

$O(b^s)$

۳. جستجوی عمقی افزایشی (Iterative Deepening Search - IDS)



این روش ترکیبی از جستجوی عمقی (DFS) و جستجوی سطحی (BFS) است که در آن جستجوی عمقی با افزایش تدریجی عمق انجام می شود تا زمانی که راه حل پیدا شود.

- ابتدا یک DFS با محدودیت عمق ۱ اجرا کن. اگر راه حلی پیدا نشد...
- یک DFS با محدودیت عمق ۲ اجرا کن. اگر راه حلی پیدا نشد...
- یک DFS با محدودیت عمق ۳ اجرا کن. و همین طور ادامه بده...

این روش محاسبات redundant دارد. چرا که در هر مرحله، DFS از ابتدا اجرا می شود.

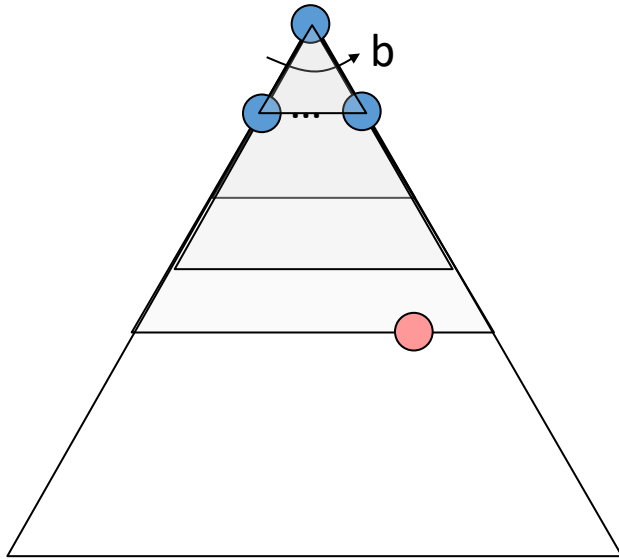
- ولی تعداد دفعاتی که هر گره بازدید می شود در مقایسه با تعداد کل گره ها در آخرین عمق جستجو بسیار کم است.

✓ نهایتاً حجم محاسبات کارهای اضافه $O(b^{s-1})$

✓ و محاسبات در عمق آخر $O(b^s)$

ارزیابی جستجوی عمقی افزایشی

اگر عمق کم عمق ترین راه حل را s در نظر بگیریم:



1. کامل بودن (Complete):

▪ بله، مانند جستجوی سطحی

2. بهینه بودن (Optimal):

▪ بله، مانند جستجوی سطحی

3. پیچیدگی زمانی (Time complexity):

▪ نمایی. مانند جستجوی عمقی

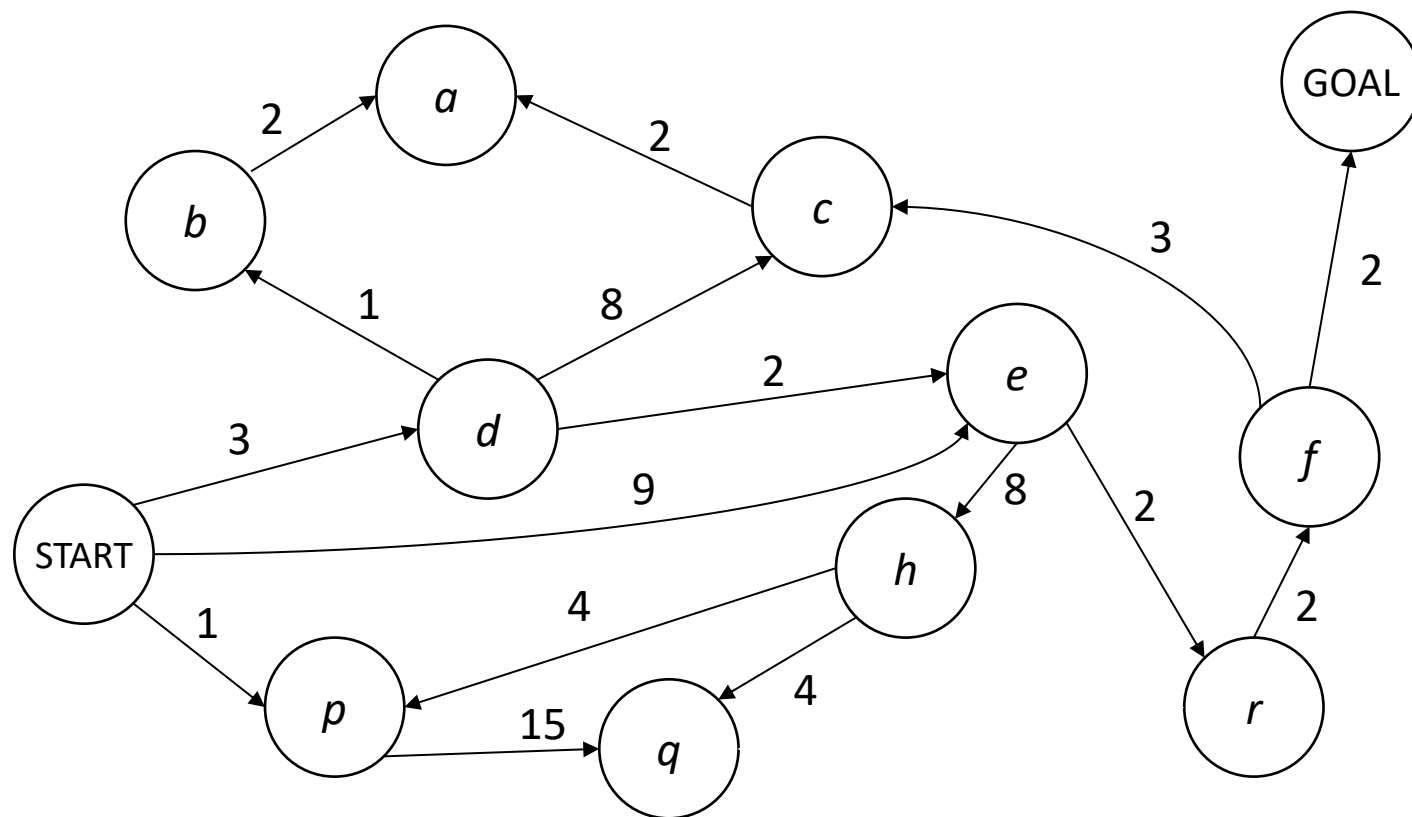
$$sb^1 + (s-1)b^2 + (s-2)b^3 + \dots + 1b^s \in O(b^s)$$

1. پیچیدگی حافظه (Space complexity):

▪ خطی - مانند جستجوی عمقی

$$O(bs)$$

جستجوی حساس به هزینه (Cost-Sensitive Search)

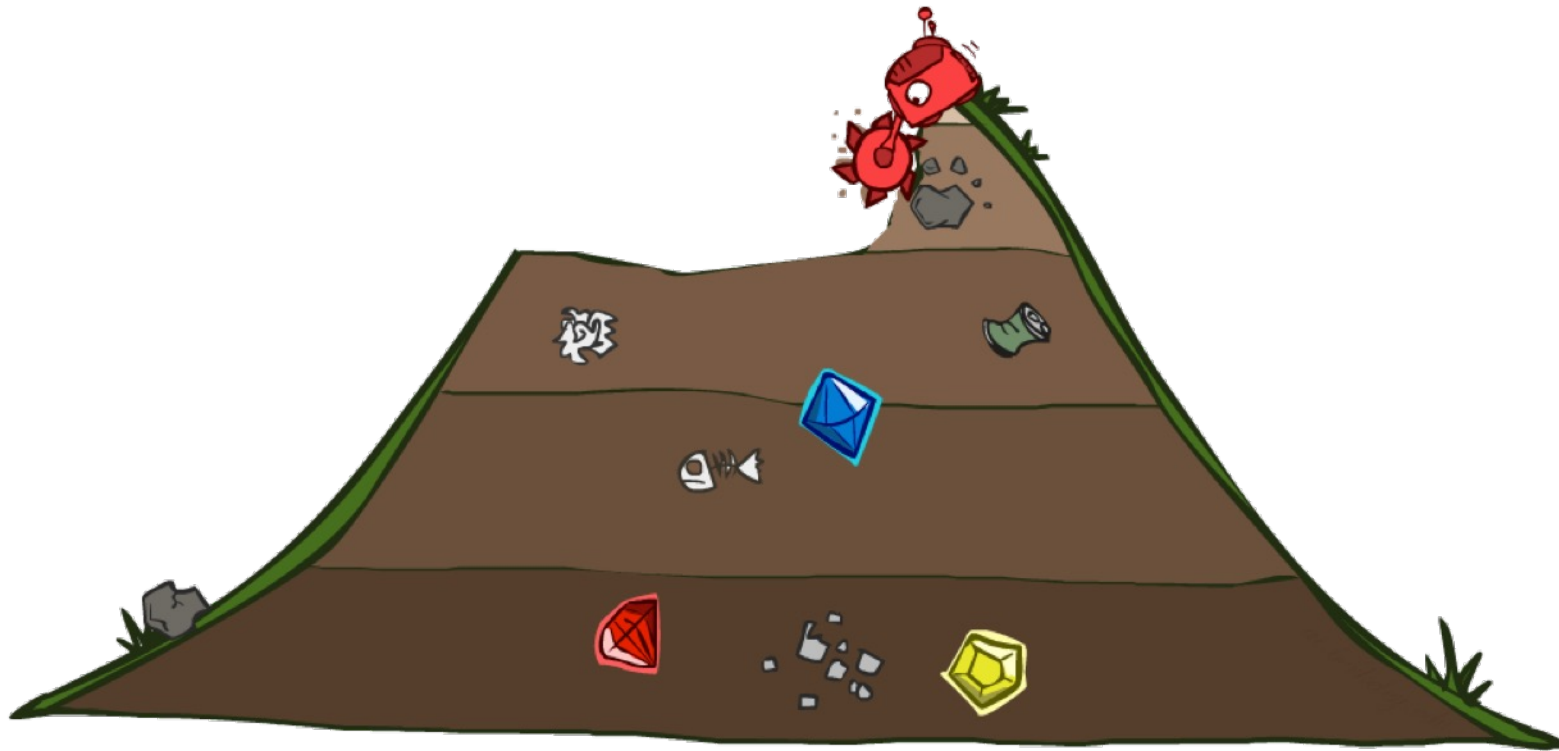


BFS کوتاه‌ترین مسیر را از نظر تعداد action پیدا می‌کند.

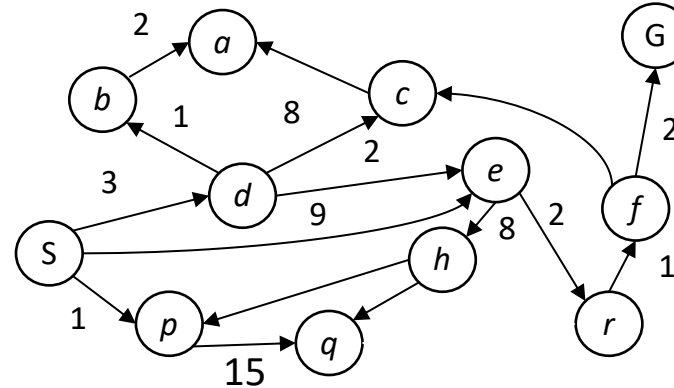
اما کم‌هزینه‌ترین مسیر را پیدا نمی‌کند.

در ادامه الگوریتم مشابه‌ای را بررسی خواهیم کرد که کم‌هزینه‌ترین مسیر را پیدا می‌کند.

۴. جستجوی هزینه یکنواخت (Uniform Cost Search - UCS)



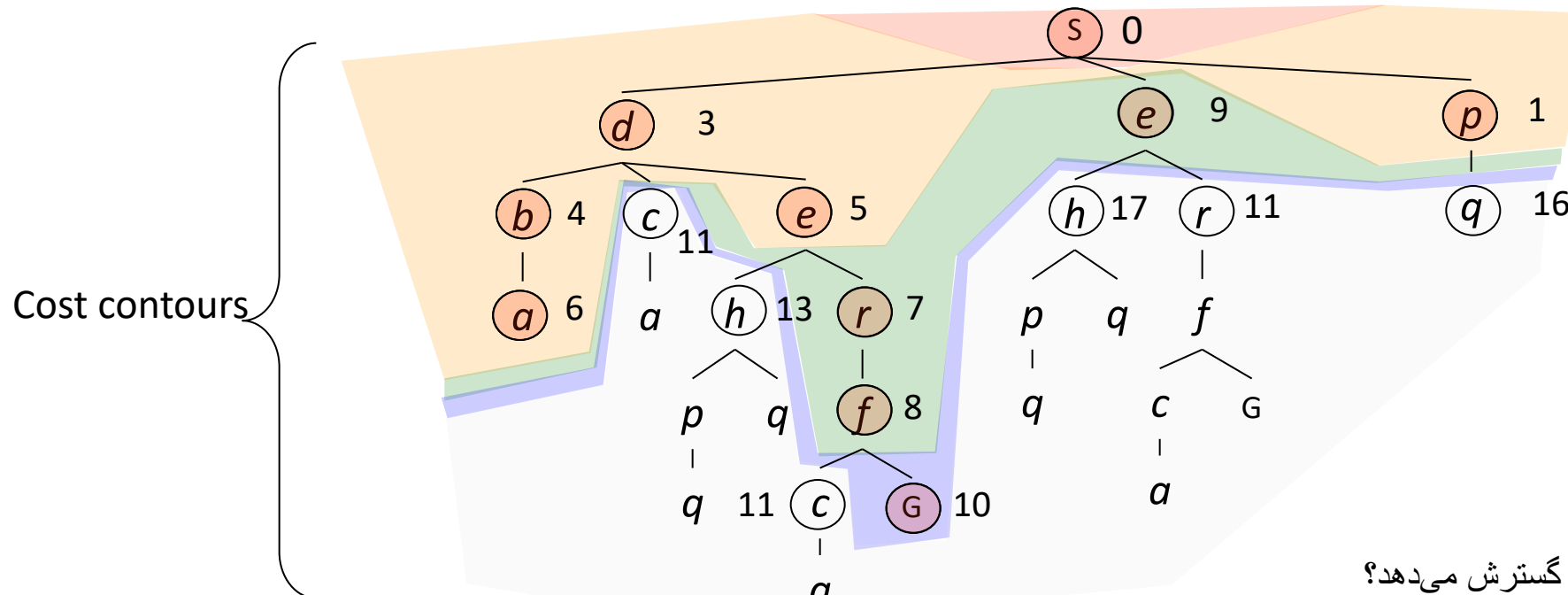
۴. جستجوی هزینه یکنواخت



$g(n)$ = هزینه از ریشه تا گره n

استراتژی: ابتدا گره‌ای که کمترین مقدار $g(n)$ دارد را گسترش بده.

Fringe یک صف اولویت است که براساس $g(n)$ مرتب شده است (اولویت: هزینه‌ی تجمعی کمتر).

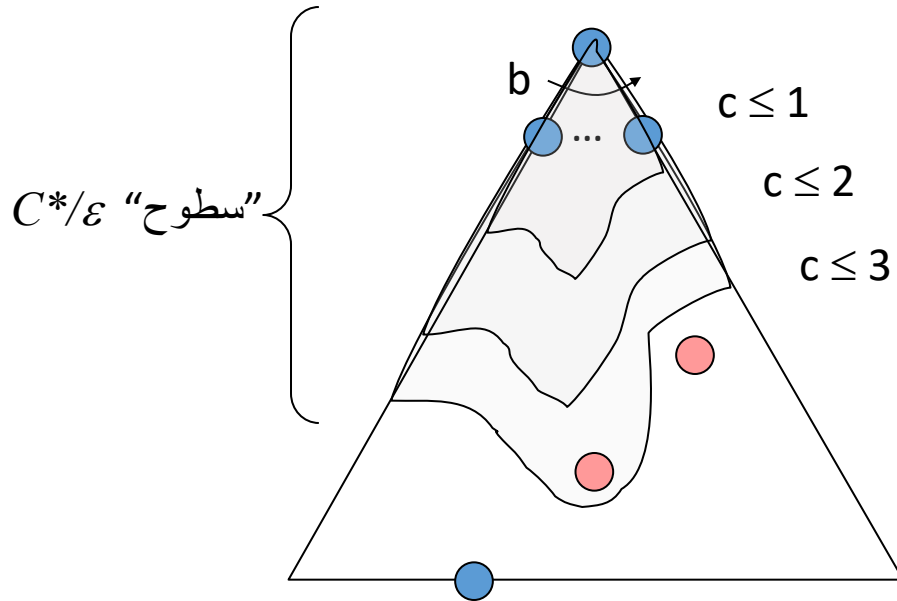


Cost contours

سوال: جستجوی هزینه یکنواخت کدام گره‌ها را گسترش می‌دهد؟

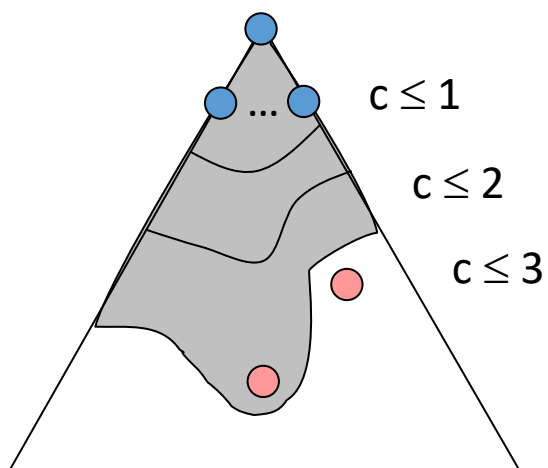
• تمام گره‌هایی که هزینه‌ی آن‌ها کمتر از نود هدف (ارزان‌ترین راه‌حل) است را پردازش می‌کند!

ارزیابی جستجوی هزینه یکنواخت



1. کامل بودن (Complete):
 ▪ بله! اگر هزینه همه عمل‌ها مثبت باشد. جوابی در عمق محدود وجود داشته باشد
2. بهینه بودن (Optimal):
 ▪ بله! اگر هزینه همه عمل‌ها مثبت باشد
3. پیچیدگی زمانی (Time complexity):
 ▪ نمایی - اگر هزینه‌ی آن راه حل C^* باشد و کمترین هزینه‌ی یال‌ها حداقل ϵ باشد،
 ▪ آنگاه "عمق مؤثر" تقریباً C^*/ϵ است.
 ▪ زمان اجرا $O(b^{C^*/\epsilon})$ است (به صورت نمایی در عمق مؤثر).
4. پیچیدگی حافظه (Space complexity):
 ▪ نمایی - تقریباً شامل آخرین سطح گره‌ها است، پس $O(b^{C^*/\epsilon})$ فضا می‌گیرد.

مشکلات جستجوی هزینه یکنواخت

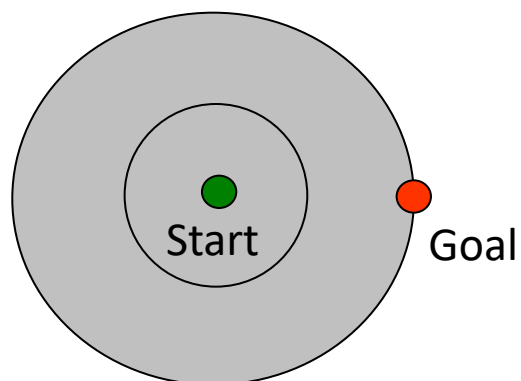



مزیت: 

UCS کامل و بهینه است!

عیب: 

گزینه‌ها را در همه‌ی جهات بررسی می‌کند.
هیچ اطلاعاتی درباره‌ی موقعیت هدف ندارد.



SOON  به‌زودی این مشکل را حل خواهیم کرد!

خلاصه جستجوهای ناآگاهانه

□ در این مسائل فرض شده که محیط کاملاً مشاهده‌پذیر، گسسته، قطعی، ایستا و ترتیبی است.

□ الگوریتم‌های جستجو دنباله‌ای از اعمال را پیدا می‌کنند که به حالت‌های هدف می‌رسند.
▪ بهینه → حداقل هزینه

□ ویژگی‌های الگوریتم‌های جستجو:

- جستجوی عمقی (Depth-first): غیرکامل، غیربهینه، مصرف حافظه کم
- جستجوی سطحی (Breadth-first): کامل، (غیر) بهینه، مصرف حافظه زیاد
- جستجوی عمقی افزایشی (Iterative deepening): کامل، (غیر) بهینه، مصرف حافظه کم
- جستجوی هزینه-یکنواخت (Uniform-cost): کامل، بهینه، مصرف حافظه زیاد

خلاصه ارزیابی جستجوهای ناآگاهانه

- تعاریف:
- **b**: فاکتور انشعاب (فرض کنید مقدار آن محدود است)
 - **m**: بیشترین عمق درخت جستجو
 - **s**: کمترین عمق رامحل (فرض کنید مقدار آن محدود است)
 - **C***: هزینه رامحل بهینه (فرض کنید مقدار آن محدود است)
 - ϵ : حداقل هزینه بین دو گره

روش جستجو	استراتژی مدیریت حاشیه (Fringe)	کامل بودن	بهینه بودن	پیچیدگی زمانی	پیچیدگی حافظه
جستجوی عمقی (DFS)	پشته (Stack)	خیر [جستجوی درختی] بله [جستجوی گرافی محدود] خیر [جستجوی گرافی نامحدود]	خیر	$O(b^m)$	$O(bm)$
جستجوی سطحی (BFS)	صف (Queue)	بله	خیر (مگر اینکه تمام هزینه‌ها برابر باشند)	$O(b^s)$	$O(b^s)$
جستجوی عمقی افزایشی (IDS)	پشته (مشابه DFS)	بله (مشابه BFS)	خیر (مشابه BFS)	$O(b^s)$ (مشابه BFS)	$O(bs)$ (مشابه DFS اما با کمترین طول مسیر)
جستجوی هزینه یکنواخت (UCS)	صف اولویت‌دار بر اساس g	بله (با فرض مثبت بودن هزینه یال‌ها)	بله (با فرض مثبت بودن هزینه یال‌ها)	$O(b^{C^*/\epsilon})$	$O(b^{C^*/\epsilon})$