بخش سوم

جستجوی ناآگاهانه

(Blind Search)

۱ـ جستجوی اول سطح (جستجوی سطحی) Breadth-first Search:

در این روش ابتدا node ریشه گسترش می یابد. اگر ریشه جواب مسئله بود الگوریتم پایان میپذیرد در غیر اینصورت فرزندان ریشه گسترش میابند. حال بصورت سطحی در بین فرزندان ریشه بررسی میکنیم که آیا به جواب رسیده ایم یا خیر؟ اگر در این سطح به جواب نرسیم تمام گره هایی که توسط ریشه تولید شده اند خودشان گسترش می یابند و سطح بعدی را تشکیل میدهند. در هر سطح جستجو برای جواب انجام میگیرد تا نهایتا به جواب برسیم.

الگوریتم جستجوی اول سطح:

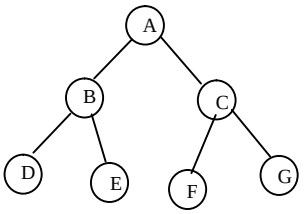
۱ـ یک صف خالی ایجاد کن و حالت اولیه را در آن قرار بده

۲ـ اگر لیست خالی است جواب نداریم در غیر این صورت عنصر سر صف را بخوان

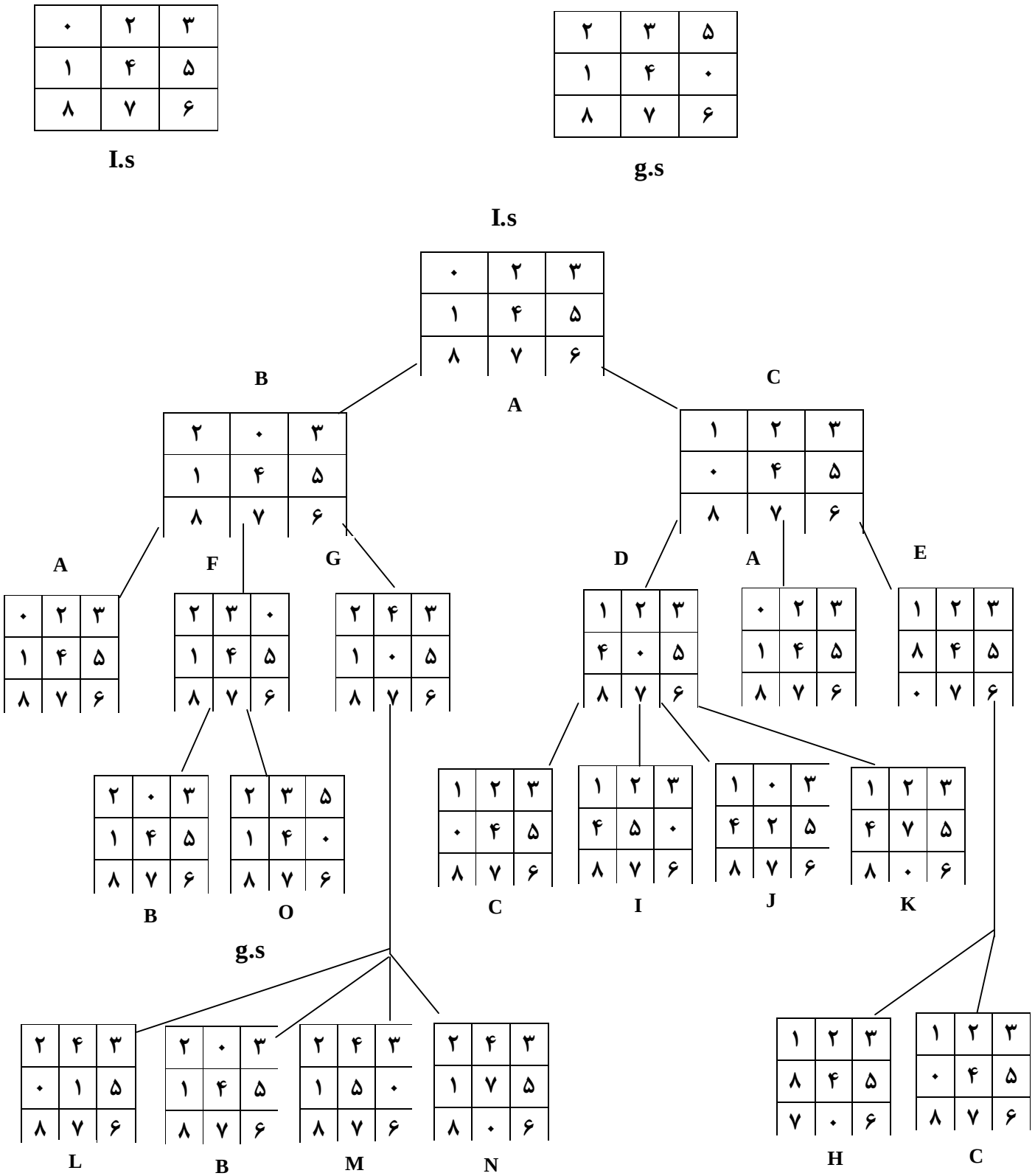
۳ـ اگر عنصر خوانده شده جواب است مسیر را به عنوان جواب برگردان

۴ـ در غیر این صورت عنصر خوانده شده را از صف خارج کن و فرزندان آن را در صورت وجود و ملاقات نشدن در انتهای صف قرار بده و به مرحله ۲ برو.

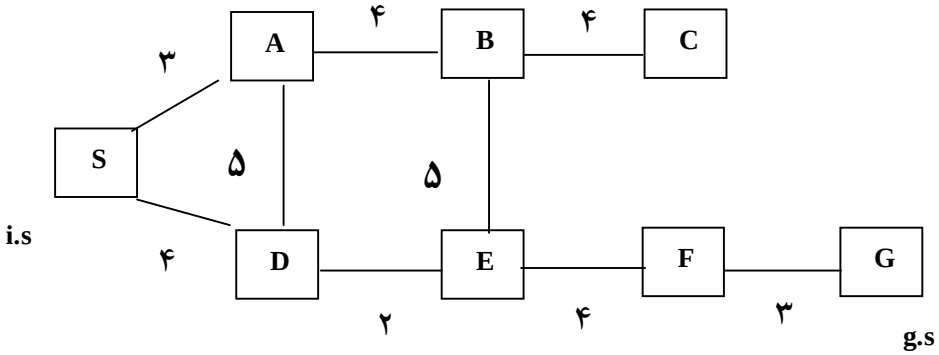
مثال:



مثال:



مثال:



ویژگی‌ها

الف) اگر راه حلی وجود داشته باشد جستجوی اول سطح ضمانت می‌کند که حتماً آن را بیابد (شرط کامل بودن)

ب) اگر چندین راه حل وجود داشته باشد جستجوی اول سطح ضمانت می‌کند که کم عمق‌ترین جواب را مشخص کند (بهینه بودن)

معایب: پیچیدگی زمانی و فضایی زیادی دارد (اگر نرخ شاخه شدن b باشد).

$$1+b+b^2+...+b^d$$

زمانی $o(b^d)$ فضا $s(b^d)$

عمق	نودها	زمان	فضا
0	1	1 ms	100
2	111	100 ms	11 k
14	10^{14}	3500 سال	11/111ترابایت

۲- جستجو با هزینه یکسان (uniform cost search)

این الگوریتم بهینه شده روش اول سطح است و در آن گره ای ابتدا توسعه داده می شود که هزینه رسیدن به آن حداقل باشد برای پیاده سازی این جستجو از صف اولویت دار استفاده می کنیم . در اول صف همیشه گره ای قرار می گیرد که هزینه رسیدن به آن حداقل بوده است.

الگوریتم

۱ـ یک صف خالی اولویت دار ایجاد کن و حالت اولیه را در آن قرار بده.

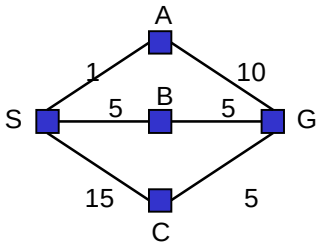
۲ـ اگر لیست خالی است جواب نداریم در غیر این صورت عنصر سر صف را بخوان.

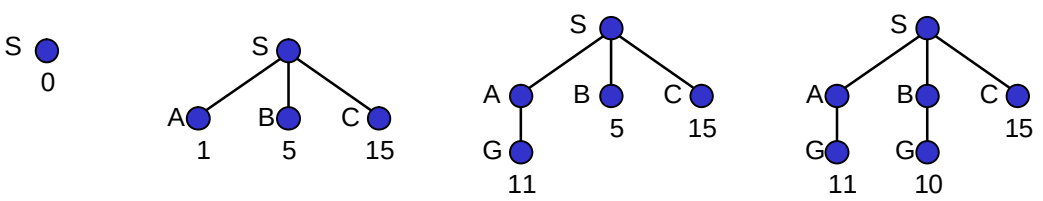
۳ـ اگر عنصر خوانده شده جواب است مسیر را به عنوان جواب برگردان.

۴ـ در غیر این صورت عنصر خوانده شده را از صف خارج کن و فرزندان آن را در صورت وجود و ملاقات نشدن براساس اولویت در صف قرار بده.

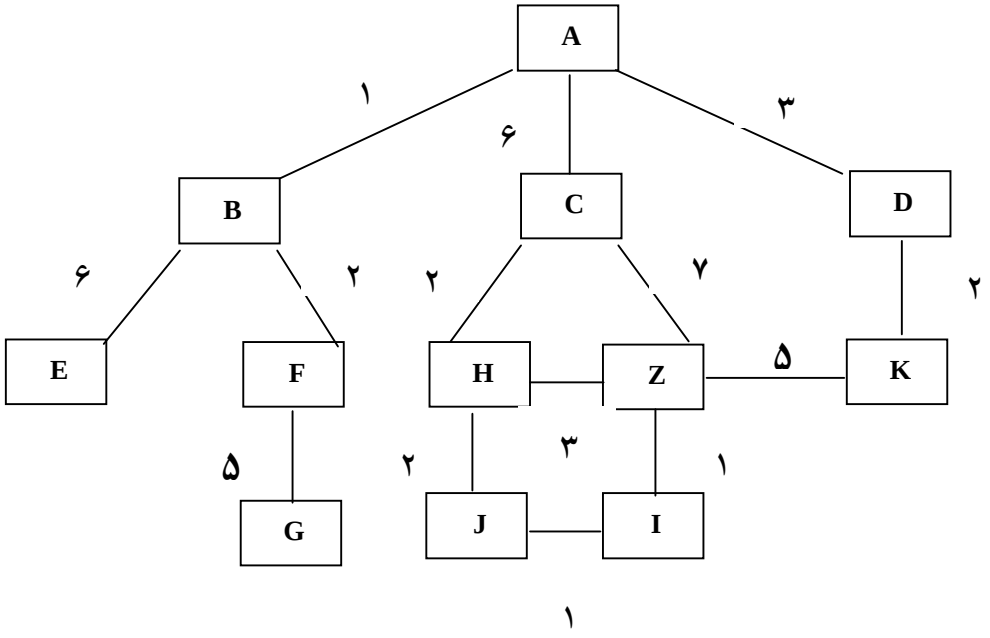
در این روش اگر به جواب برسیم اولین جواب ارزانترین جواب است.

مثال: می‌خواهیم با روش UCS از S به G برسیم.





تمرین:



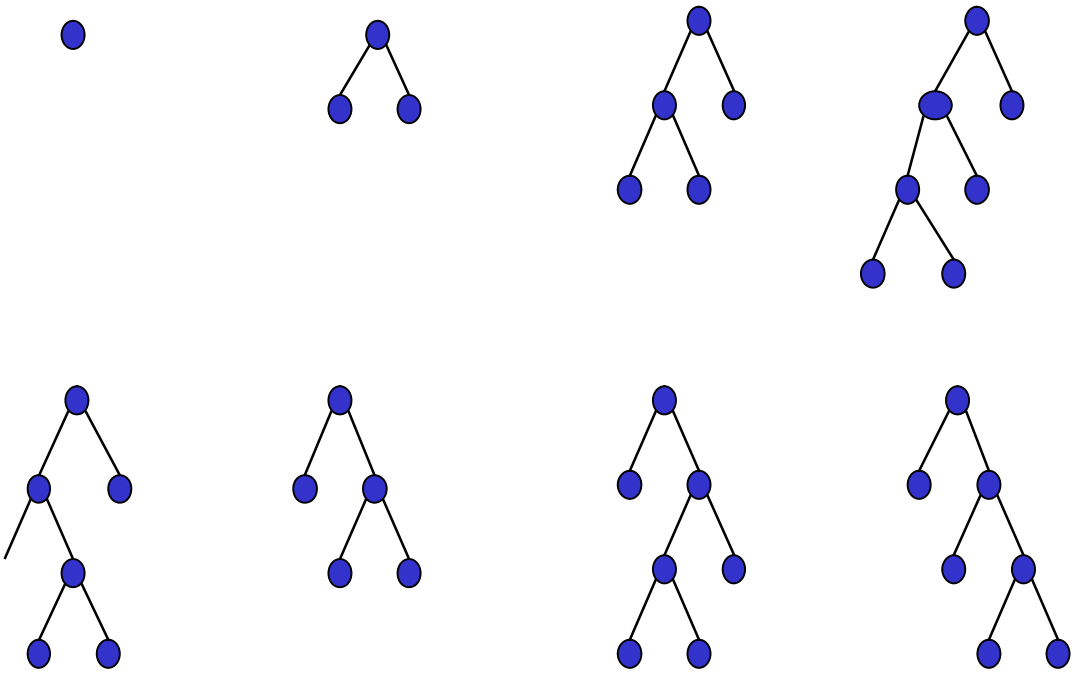
ویژگی ها

۱ـ زمانی که شرایط عمومی برقرار باشد اولین راه حل پاسخ ضمانت می‌کند که ارزانترین راه می‌باشد. اگر بعضی از عملگرها هزینه منفی داشته باشند باید جستجویی از تمام گره‌ها صورت گیرد، تا راه حل بهینه پیدا شود (در صورتی این راه حل بهینه است که هزینه منفی نداشته باشیم).

۲- پیچیدگی زمانی و فضا مانند حالت قبل است.

۳- روش جستجوی اول ـ عمق Depth-first Search

در این روش همیشه یکی از گره هایی که در پایین ترین عمق قرار دارد بسط داده می شود و این کار را آن قدر ادامه می دهیم تا به جواب برسیم در صورت رسیدن به بن بست مسیر دیگری انتخاب می شود (Backtracking) این استراتژی جستجو را می توان به وسیله صف پیاده سازی کرد که همیشه حالات جستجو شده جدید در جلوی صف قرار می گیرد .



الگوریتم

۱ـ صف خالی ایجاد کن و نود اولیه را در آن قرار بده.

۲ـ اگر صف خالی است مسئله جواب ندارد در غیر این صورت عنصر سر صف را بخوان.

۳ـ اگر عنصر خوانده شده جواب است مسیر را به عنوان جواب برگردان.

۴ـ در غیر این صورت عنصر سر صف را خارج کن(از pop stack کن) و فرزندان آن را در صورت رویت نشدن به ابتدای صف اضافه کن (Push) و به مرحله ۲ برو.

ویژگی ها

۱ـ نیاز به حافظه کمی دارد چرا که نیاز به ذخیره مسیر واحدی از ریشه به یک گره برگی دارد به علاوه برخی گره‌های بست داده نشده.

جستجوی عمقی چون فضای کمی را به دنبال جواب می‌گردد پس شانس خوبی برای یافتن جواب دارد برای مسائلی که راه حل زیادی دارد جستجوی اول عمق سریع‌تر از جستجوی اول سطح عمل می‌کند. جستجوی اول عمق ممکن است، هنگام پایین رفتن در یک مسیر اشتباه گیر کند این مسئله در مسائل عمیق و نامحدود بیشتر بروز می‌کند و ممکن است هیچ‌گاه الگوریتم به جواب نرسد و یا راه حلی را که پیدا می‌کند طولانی‌تر از راه حل بهینه باشد. پس جستجوی اول عمق نه کامل است و نه بهینه.

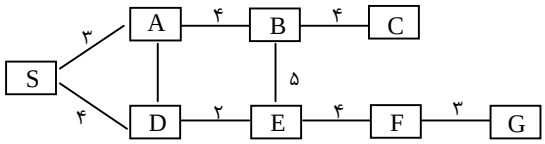
۲- اگر نرخ شاخه شدن b بوده و M عمق جواب باشد.

مقدار حافظ مصرفی $\Leftarrow S(b.M)$

پیچیدگی زمانی در بدترین حالت (b^M)

مثال:

با استفاده از الگوریتم اول عمق، از حالت S به حالت G برسید.



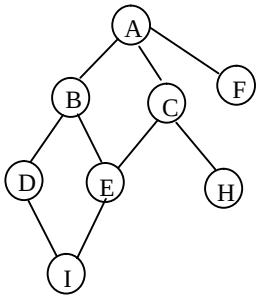
۱ـ ابتدا سمت راست بعد چپ

۲ـ به ترتیب حروف الفبا

۳ـ براساس هزینه مسیر

مثال:

اگر در گراف زیر جستجوی اول عمق را از رأس C شروع کنیم کدام گره‌ها به ترتیب از چپ به راست رویت می‌شود. فرض کنید فرزندان یک گره براساس ترتیب حروف الفبا انتخاب شوند.



۱) ABCDEFHI

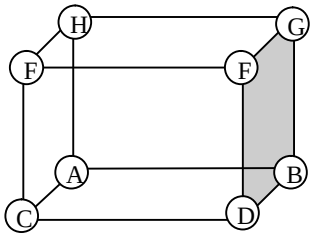
۲) CABDIEFH

۳) CAEHBFID

۴) CABDEHIF

مثال:

پیمایش اول عمق برای گراف مقابل با شروع از گز A کدام است؟ (برحسب حروف الفبا)



۱) ABDCEFHG

۲) ACEFDBHG

۳) ACDFBEGH

۴) AHGBDCEF

مثال:

با روش اول عمق، از **i.s.** به **g.s** بروید. روش از راست به چپ

۱	۲	۳
۸	۴	۵
۷	۰	۶

	۲	۳
۱	۴	۵
۸	۷	۶

الگوریتم روش عمقی

1. add the root to the open set.

2. unttil open set in empty or the Goal has been reached.

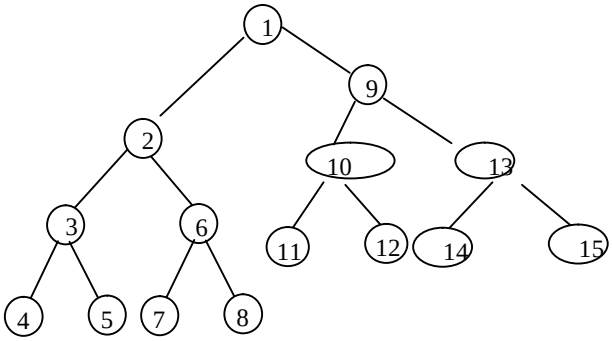
2a. remove the first node from the open set.

2b. if this node is not the Goal there add any children that haven't been visited to the front of the open set

3. Goal found.

۴- روش جستجوی عمقی محدود شده *Depth-Limited Search*

این روش از بدام افتادن جستجوی عمقی توسط به کارگیری یک برش روی عمیق ترین مسیر جلوگیری می کند یعنی جستجو را از یک حداکثر عمق بیشتر ادامه نمی دهیم و مقدار آن توسط طراح مشخص می گردد . این الگوریتم زمانی مناسب است که ما حدود فاصله اولیه از جواب را می دانیم و تعداد جواب برای مسئله وجود داشته باشد .



الگوریتم

۱ـ یک صف خالی ایجاد کند و حالت اولیه را در آن قرار بده.

۲ـ اگر صف خالی است مساله جواب ندارد وگرنه عنصر سر صف را بخوان.

۳ـ اگر عنصر خوانده شده جواب است مسیر را به عنوان جواب اعلام کن.

۴ـ عنصر خوانده شده را از صف خارج کن.

۵ ـ اگر عمق عنصر خوانده شده کوچکتر از L است به مرحله ۶ برو و در غیر این صورت به مرحله ۲ برو.

۶ ـ فرزندان عنصر خوانده شده را در جلوی صف قرار بده (در stack) و به مرحله ۲ برو.

ویژگی ها

۱ـ جستجوی عمقی محدود شده به شرطی که عمق انتخابی خیلی کوچک نباشد کامل است اما بهینه نیست پیچیدگی زمانی و پیچیدگی فضا مانند روش اول عمق است.

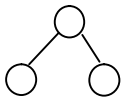
$S\ (b.L)$

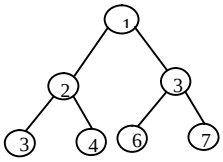
$O\ (b^L)$

۵ ـ جستجوی عمیق‌کننده تکراری Iterative-deepening search

این جستجو از مزایای هر دو روش اول سطح و اول عمق استفاده مینماید. این روش برای رفع مشکل انتخاب مقدار عمق مناسب در روش قبلی به وجود آمده است در این روش محدوده عمقی مناسب توسط امتحان نمودن تمامی محدوده عمق‌ها به دست می‌آید ابتدا عمق صفر سپس عمق یک و الی آخر. در هر عمق، به روش اول عمل عمق جستجو را انجام میدهیم اگر به جواب نرسیدیم آنگاه عمق را افزایش داده و از اول جستجو را به روش اول عمق ادامه میدهیم.

$d = 0$

$d = 1$ 

$d = 2$ 

نسبت به جستجوی سطحی زمان بیشتری را می‌گیرد.

الگوریتم

algoritm

for $c = f$ *to* $c = \infty$

if depth limited search with c succeeds there return result endif

return failure

ویژگی‌ها

۱ـ این روش مزایای جستجوی عمقی و سطحی را با هم ترکیب می‌کند.

۲ـ این روش هم کامل است و هم بهینه

۳ـ از نظر حافظه مصرفی همانند جستجوی اول عمق بوده و برابر $S(b.d)$ است.

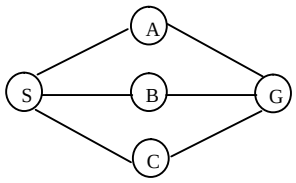
۴ـ در این روش چون در هر مرتبه مجدد از ریشه شروع می‌کنیم و دوباره تمام درخت را بسط می‌دهیم پس گره‌هایی که در پایین‌ترین سطح هستند یک بار بسط داده می‌شوند آنهایی که در یک سطح بالاترند دو بار بسط داده می‌شوند و گره ریشه $d+1$ بار بسط داده می‌شود. بنابراین مجموع دفعات بسط در این جستجو برابر است با:

$$1(d+1) + db + (d-1)b^2 + \ldots + 3b^{d-2} + 2b^{d-1} + 1b \quad o(b^d)$$

پس پیچیدگی زمانی این جستجو برابر است با $o(b^d)$. عبارت بسط نشان می‌دهد که زمان بیشتری نسبت به روش‌های قبل تلف می‌شود.

۵ ـ در حالت کلی زمانی این الگوریتم مفید است که فضای جستجوی بزرگی وجود داشته باشد. و عمق راه نیز مجهول باشد. این روش در چنین شرایطی نسبت به روش‌های قبل برتری دارد.

مثال:



۶ ـ جستجوی دو طرفه Bidirectional Search

ایده این روش، جستجوی هم زمان از حالت شروع به هدف یا به سمت جلو و از حالت نهایی به سمت عقب می‌باشد و وقتی به یک حالت مشترک رسیدیم این مسیر جواب خواهد بود.

ویژگی‌ها

۱ـ اگر فاکتور انشعاب در دو جهت b باشد و راه حلی در عمق b وجود داشته باشد این راه حل با زمان اجرای $o(b^{\frac{d}{2}})$ پیدا خواهد شد زیرا در هر دو جهت نیمی از راه را می‌پیماید بررسی کنید پیچیدگی فضای این روش جستجو نیز ($b^{\frac{d}{2}}$) است.

معایب

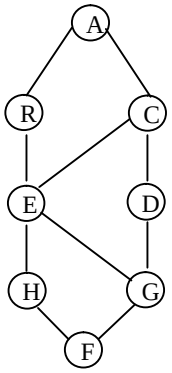
۱ـ اگر چندین هدف داشته باشیم چگونه از کدامیک به طور معکوس حرکت کنیم. در بعضی از مسایل حالت هدف تعریف مشخصی ندارد مانند بازی شطرنج؟ از چه جستجویی در هر نیمه باید استفاده شود؟

b =نرخ انشعاب L=محدوده عمق تعیین شده d=عمق جواب n=حداکثر عمق

	دو طرفه	عمیق‌کننده تکراری	عمیق محدود شده	اول عمق	هزینه یکسان	اول سطح	
زمان	$b^{\frac{d}{2}}$	b^d	b^L	B^m	b^d	b^d	
فضا	$b^{\frac{d}{2}}$	b.d	b.L	b.m	b^d	b^d	
بهینه بودن	Yes	Yes	No	No	اگر هزینه منفی وجود داشته باشد	Yes	
کامل بودن	Yes	Yes	No	No	Yes	Yes	

تمرین ۱

کدامیک از الگوریتم‌های جستجوی زیر از لحاظ زمان و حافظه روی گراف زیر بهتر عمل می‌کند.



A : گره شروع

G : هدف

۱ـ اول سطح

۲ـ اول عمق

۳ـ هزینه یکسان

۴ـ جستجوی محدود شده با عمق ۲

تمرین ۲

کدامیک از گزینه‌های زیر روش‌های جستجو را از بهترین به سمت نامناسب‌ترین بررسی نشان می‌دهد (از سمت چپ). معیار مقایسه را به ترتیب اولویت‌های بهینه بودن، کامل بودن، مرتبه زمانی و مکانی جستجو در نظر بگیرید.

۱ـ اول عمق ـ اول سطح

۲ـ عمقی مکرر ـ عمقی محدود شده ـ اول عمق

۳ـ جستجو با هزینه یکسان ـ جستجوی دو طرفه

۴ـ هیچ‌کدام