
بسم الله الّرحمن الّرحیم

موضوع تحقیق:روش های جستجو در هوش مصنوعی

روش های جستجو هوش مصنوعی

ما به عنوان یک عامل هوشمند، در مسائل روزمره ای که برایمان پیش می آید با اسفاده از تکنیک های جستجو اقدام به یافتن راه حلی برای مسئله می کنیم. به عنوان مثال خانه ای را فرض کنید که در آن به دنبال شی خاصی هستیم. همچنین فرض کنید هیچ اطلاعی در مورد خانه و محل قرارگیری اشیا در خانه نداریم. در چنین مواردی برای یافتن شی موردنظر، خانه را اتاق به اتاق و در هر اتاق همه بخش های آن اتاق را جستجو می کنیم. نتیجه جستجو تنها در صورتی می تواند به دست آید که جواب مسئله در فضایی که آن را جستجو می کنیم، موجود باشد.به عنوان یک مثال دیگر فرض کنید می خواهیم از تبریز به سمت شیراز حرکت کنیم. هدف ما نیز رسیدن به شیراز در کمترین زمان ممکن می باشد. یکبار دیگر نیز ما به عنوان یک عامل هوشمند، با در دست داشتن نقشه راه های کشور براحتی مسیر حرکت خود را مشخص می کنیم. به طور حتم هنگام جستجو بر روی نقشه برای یافتن کوتاهرین مسیر حرکت از تبریز به شیراز، از جستجوی مسیر حرکت تبریز–ارومیه–شیراز صرف نظر می کنیم. حال سوالی که اینجا مطرح می شود اینست از به جستجوی راه حل در مسیر تبریز–ارومیه–شیراز نمی پردازیم.مهمترین تفاوت دو مثال ذکر شده را ناآگاهانه بودن جستجو در مثال اول و آگاهانه بودن جستجو در مثال دوم می باشد. بدین معنی که در مثال اول ما از ابتدا کل فضای جستجو را برای یافتن جواب مسئله جستجو می کنیم و در واقع

برای حرکت سریع تر در فضای جستجو (حرک بین اتاق ها و بخش های مختلف هر اتاق) از آگاهی خاصی بهره مند نیستیم. در نقطه مقابل و در مثال دوم برای حرکت در فضای جستجو به منظور یافتن کوتاهترین مسیر از تبریزبه شیراز، از روش خاصی برای انتخاب شهر بعدی استفاده می کنیم. به عبارت دیگر در مثال دوم علاوه بر اینکه مسیری از مبدا به مقصد پیدا می کنیم، همچنین کوتاهی مسیر نیز برایمان از اهمیت ویژه ای برخوردار می باشد. در واقع عملی که برای سنجش کوتاهی مسیر انجام می دهیم موجب آگاهانه بودن جستجوی ما در فضای جستجو می شود. در مثال دوم از تابعی استفاده می کنیم که فاصله شهر انتخابی تا مقصد را برایمان **تخمین** می زند که به آن تابع هیوریستیک یا تابع مکاشفه خواهیم گفت.

حال ممکن است این تصویر به وجود آید که جستجوی آگاهانه بهتر از جستجوی ناآگاهانه می باشد. اما چنین تصوری را نمی توان قطعا درست تلقی کرد. چراکه در صورت ناآگاهانه بودن ماهیت مسئله چاره جز جستجوی ناآگاهانه نخواهیم داشت. همچنین سریعتر بودن جستجوی آگاهانه نسبت به جستجوی ناآگاهانه نیز بستگی به شرایط مسئله خواهد داشت و موارد بسیاز زیادی می توان یافت که در آن جستجوی ناآگاهانه سریعتر از جستجوی آگاهانه ما را به جواب مسئله می رساند.

در حالت کلی روش های جستجو را می توان به سه دسته زیر تقسیم کرد

- ناآگاهانه
- هیوریستیکی
- متاهیوریستیکی

مسائلی وجود دارند که فضای جستجوی آنها به اندازه ای بزرگ می باشد که استفاده از روش های جستجوی ناآگاهانه و هیویریستیکی را برایمان غیرممکن می سازد. در چنین مواردی از روش های متاهیوریستیکی یا فرامکاشفه ی استفاده خواهیم کرد.

انواع مسائل در هوش مصنوعی

مسائل مطرح شده در هوش مصنوعی را از چند دیدگاه مختلف می توان مورد بررسی قرار دارد :

- هدف یافتن تنها یک جواب برای مسئله است یا همه جواب های ممکن برای مسئله باید جستجو شوند؟
- پاسخ دهی الگوریتم جستجو برای حل مسئله بلادرنگ خواهد بود یا زمان بیشتری برای حل مساله می توان اختیار کرد
- آیا در مسئله محدودیت هایی وجود دارد یا نه ؟

همه اینها سوالاتی هستند که هنگام طراحی الگوریتمی برای پویش در فضای جستجو باید به آنها توجه کنیم. به عنوان مثال در صورتی که همه جواب های مسئله مدنظر ما باشد، در اینصورت حتی در صورت آگاهانه بودن ماهیت مسئله نیز نمی توان از روش های هیوریستیکی استفاده کرد. به عنوان مثال در صورتی که هدف یافتن همه مسیرهای ممکن از تبریز به شیراز باشد، استفاده از هیوریستیک تنها موجب پیچیده شدن الگوریتم و حتی موجب ناقص بودن جواب های مسئله خواهد شد.

سیستم های پویا سیستم هایی هستند که در آنها به طور مداوم ورودی ها و حالت مسئله در حال تغییر است و هربار تغییر در یکی از پارامترها موجب خواهد شد که عمل جستجوی مجددی در فضای جستجو انجام پذیرد. در چنین مواردی ممکن است مدت زمان پاسخ دهی سیستم برای حل مساله بسیار کم باشد. همچنین ممکن است سیستم مدت زمان کافی برای حل مساله در اختیار داشته باشد که هریک از این موارد تاثیر بسزایی در طراحی الگوریتم جستجو خواهند داشت.

فرض کنید می خواهیم در یک صفحه شطرنج استاندارد، 8 وزیر را به گونه بر روی صفحه شطرن قرار دهیم که هیچیک از آنها وزیر دیگری را گارد ندهند. در این مسئله که به مسئله 8 وزیر معروف است، محدودیتی وجود دارد که بر اساس آن هیچ وزیری باید همدیگر را گارد ندهند. به صورت تجربی می توان چنین گفت که بیشتر مسائلی که با آنها سروکار داریم دارای محدویت هایی در مسئله هستند. به چنین مسائلی مسائل ارضای محدودیت نیز می گوییم. وجود محدودیت در مسئله قدری موجب پیچیدگی مسئله می شود اما در اکثر موارد نیز موجب خواهد شد فضای جستجو مسئله بسیار کوچکتر از حالتی باشد که در آن مسئله هیچ محدودیتی در خود ندارد.

مسائلی که در عمل با آنها مواجه هستیم ترکیبی از موارد فوق را در خود دارند. به عنوان مثال در مسئله ی 8 وزیر ، مسئله دارای محدودیت بوده ولی در مقابل هدف یافتن تنها یک جواب در مدت زمان کافی است. و در واقع یافتن جواب در آن نیازی به بلادرنگ بودن ندارد. در ادامه ، به بررسی روش های جستجو ناآگاهانه و نحوه پیاده سازی آنها خواهیم پرداخت.

الگوریتم های جستجوی ناآگاهانه

روش های جستجوی ناآگاهانه ای که در این بخش با آنها آشنا خواهیم شد، به قرار زیر هستند :

- جستجوی عمقی
- جستجوی سطحی
- جستجوی عمقی محدود شده
- جستجوی عمقی تکرار شونده

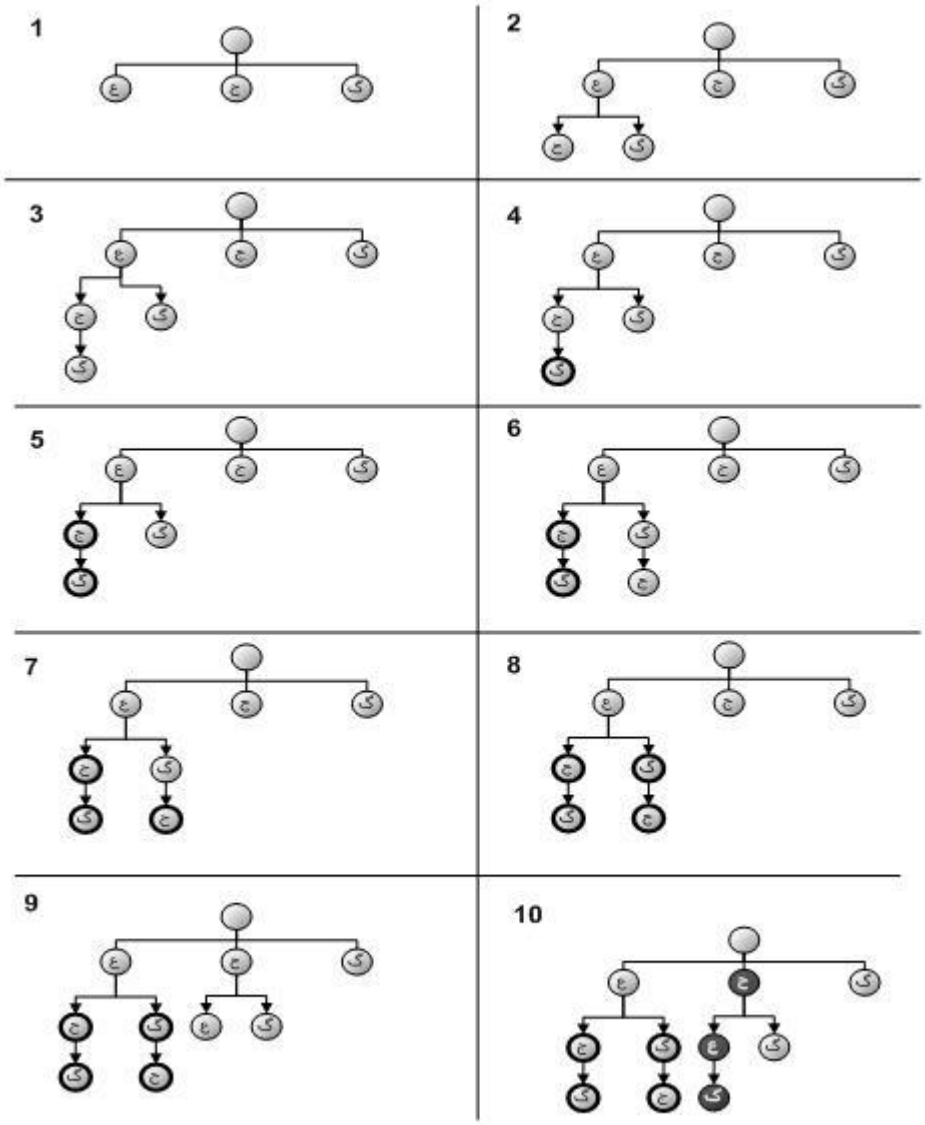
در همه روش های جستجو تابعی به نام تابع گسترش وجود دارد که وظیفه این تابع تولید فرزندان یک گره می باشد. تنها تفاوت روش های جستجوی فوق در نحوه پیمایش فضای جستجو خواهد بود. مسلم است که مرتبه زمانی و مکانی هریک از این روش ها برای یک مسئله مشخص متفاوت از یکدیگر خواهد بود. در حالت کلی

مقایسه روش های جستجو برای یک مسئله و انتخاب روش جستجوی مناسب برای آن با استفاده از فاکتورهای زیر تعیین می شود :

- فاکتور انشعاب : حداکثر تعداد فرزندانی که در هر سطح از گراف وجود خواهد داشت
- عمق درخت : حداقل عمقی که در آن عمق از گراف جوابی برای مسئله پیدا خواهد شد

جستجوی عمقی

از اسم این روش جستجو پیداست که پیمایش گراف فضای جستجو با حرکت در عمق انجام می پذیرد. به عنوان مثال فرض کنید با سه کلمه "جستجو" ، "گراف" و "عمقی" می خواهیم **یک** جمله معنی دار بسازیم. شکل زیر مراحل مختلف تشکیل گراف هنگام جستجوی عمقی گراف را نشان می دهد :



الگوریتم جستجو چنین است :

ابتدا فرزندان ریشه درخت تولید شده و در سطح بعدی درخت قرار می گیرند. در ابتدای کار هریک از کلمات می تواند برای شروع جمله بکار روند. سپس اولوین گره ارای کلمه "عمقی" در سطح 1 از درخت انتخاب شده و همه فرزندان قابل تولید آن در سطح 2 درخت تشکیل می شوند(شکل 2). سپس در سطح 2 از درخت گره شامل کلمه "جستجو" انتخاب شده و همه فرزندان آن در سطح 3 درخت تشکیل می شوند(شکل 3). در نهایت نیز کلمه

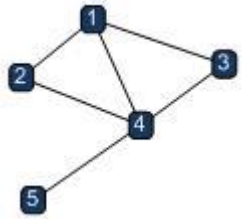
"گراف" انتخاب شده ور همه فزرندان آن تولید می شوند. با توجه به اینکه دیگر هیچ فزرندی برای این گره وجود ندارد، بنابراین جمله "عمقی جستجوی گراف" که از پیمایش درخت از ریشه تا گره فعلی به دست می آید بررسی می شود تا صیحیح بودن این جمله را تعیین کند. همانطور که می دانیم این جمله از نظر قواعدی جمله صحیحی نمی باشد. بنابراین گره فعلی از حافظه حذف شده (شکل 4) و گره "جستجو" در سطح 2 بررسی می شود تا اگر فزرند دیگری برای این گره وجود داشته باشد، ادامه جستجو از آن فرزند ادامه یابد. اما فرزند دیگری برای این گره وجود ندارد. از اینرو این گره نیز از حافظه حذف می شود (شکل 5)

حال فرزندان گره "گراف" در سطح 2 تولید می شوند (شکل 6). همانطور که می بینیم فرزند دیگری برای گره "جسجتو" در سطح 3 وجود ندارد. بنابراین این گره نیز به همراه گره پدر از حافظه حذف خواهد شد (شکل 7 و 8). سپس به سطح 1 بازگشته و فرزندان گره "جستجو" را گسترش می دهیم(شکل 9). سپس فرزندادن گره "عمقی" در سطح 2 گسترش یافته صحت جمله ساخته شده از آن کلمات بررسی می شوند. در این مرحله جمله از نظر قواعدی درست تشخیص داده شده و عبارت "جستجوی عمقی درخت" به عنوان جواب مسئله پیدا می شود (شکل 10).

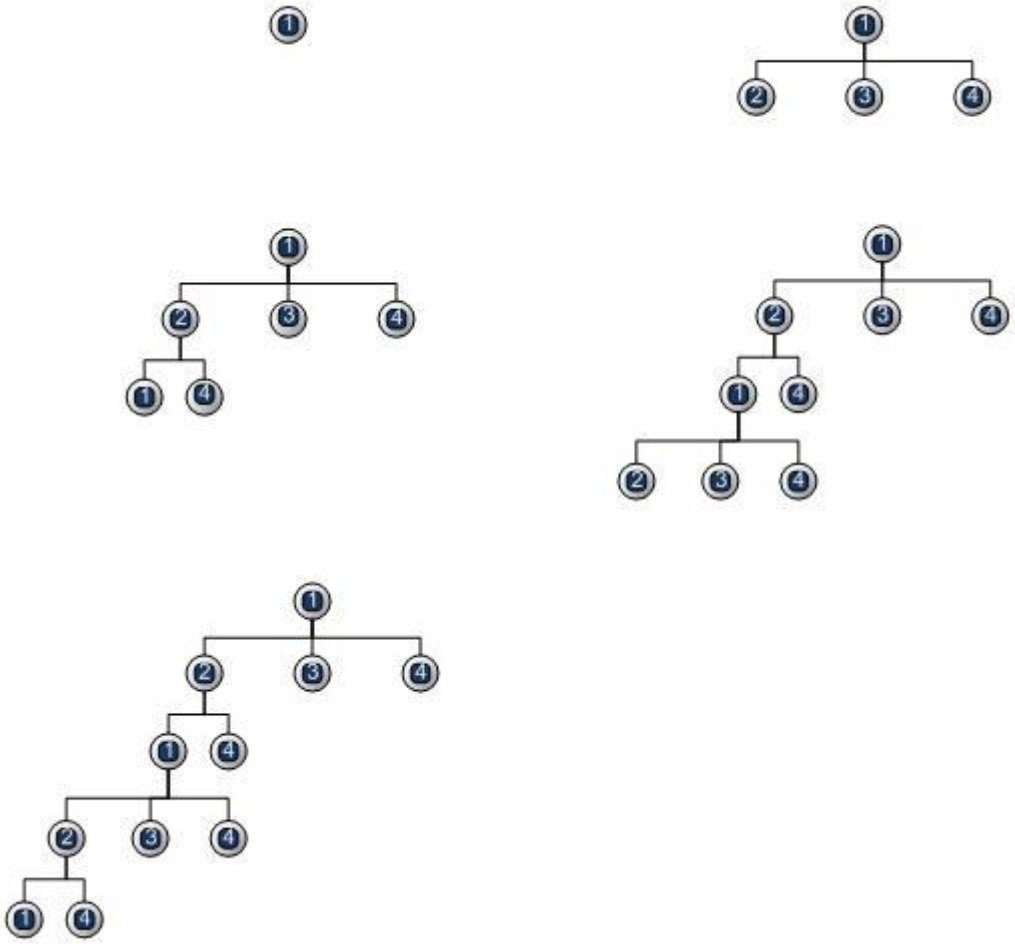
در مسئله فوق از دو تابع گسترش و تست هدف برای گسترش دادن فرزندان گره و تعیین صحت جواب مسئله استفاده کردیم. این توابع در مسائل مختلف به روش های گوناگونی پیاده سازی می شوند و الگوریتم کلی برای آن ها وجود ندارد.

جستجوی عمقی محدود شده

مهمترین مشکل روش جستجوی عمقی را که در برخی از مسائل با آن مواجه خواهیم بود، قرار گرفتن این روش جستجو در حلقه بینهایت می باشد. به عنوان مثال گراف شکل 1 را در نظر بگیرید :



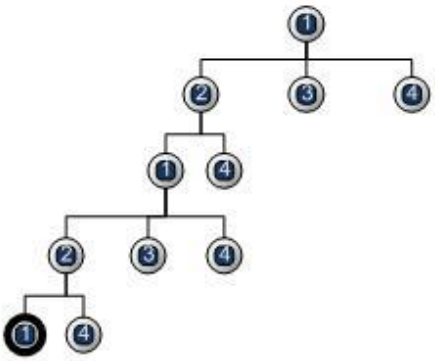
فرض کنید می خواهیم مسیری از گره 1 به گره 5 با استفاده از روش جستجوی عمقی پیدا کنیم. شکل 2 مراحل مختلف تشکیل درخت جستجو را نشان می دهد

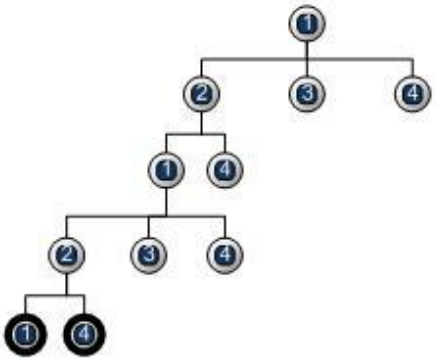


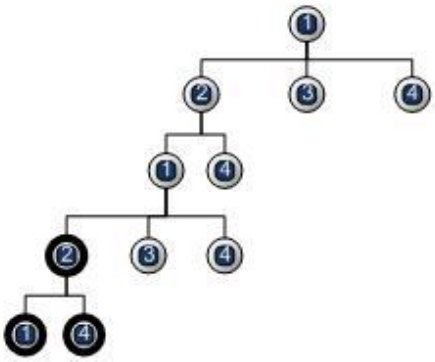
همانطور که از شکل پیداست ، استفاده از روش جستجوی عمقی برای حل این مساله هیچگاه به یافتن جواب منجر نخواهد شد. چرا که حلقه بینهایتی از گسترش گره های 1 و 2 تشکیل می شود. مشکل قرار گرفتن در حلقه بینهایت از گسترش حالت های تکراری در هنگام جستجو حاصل شده است. بنابراین یکی از روش های حل این مشکل تشخیص حالت های تکراری هنگام گسترش فرزندان یک گره می باشد.

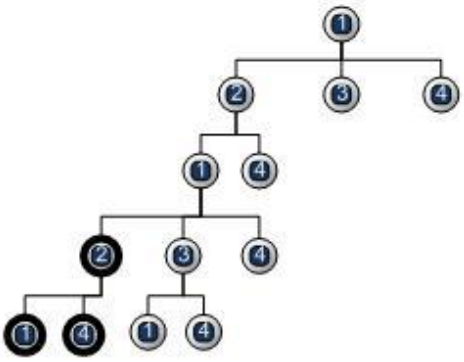
تشخیص حالت های تکراری یکی از مهمترین نکات طراحی الگوریتم های هوش مصنوعی می باشد که وابستگی بسیاری به نحوه نمایش راه حل مساله دارد. نحوه نمایش مساله و تشخیص حالت های تکراری را در انتهای فصل و در مثال های مختلف مورد بررسی قرار خواهیم داد. در حال حاضر با روش دیگری به رفع این مشکل خواهیم پرداخت.

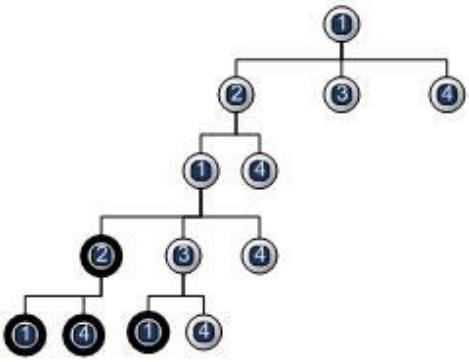
می دانیم در صورتی که اگر مسیری از گره 1 به گره 5 در گراف شکل 1 وجود داشته باشد، طول این مسیر نمی تواند بیش از 5 یال باشد. بنابراین می توان با اتخاذ یک محدودیت به روش جستجوی عمقی از گسترش گره هایی که در عمق 5 از درخت جستجو قرار دارند، جلوگیری کرد. به عبارت دیگر برای گره هایی که در عمق 5 از درخت جستجو قرار دارند ، هیچ فرزندی تولید نکنیم. پیمایش در فضای جستجوی مساله با این روش را روش جستجوی عمقی محدود شده می نامیم. یکبار دیگر با استفاده از روش جستجوی عمقی محدود شده به عمق 5 به حل مساله می پردازیم. شکل زیر مراحل مختلف تشکیل درخت جستجو را نشان می دهد (ادامه درخت جستجوی شکل 2) :

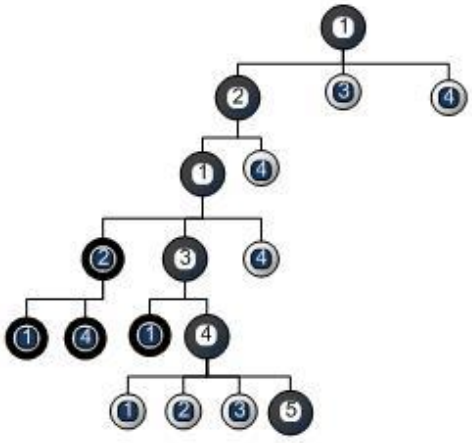








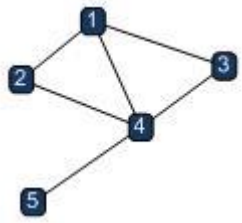




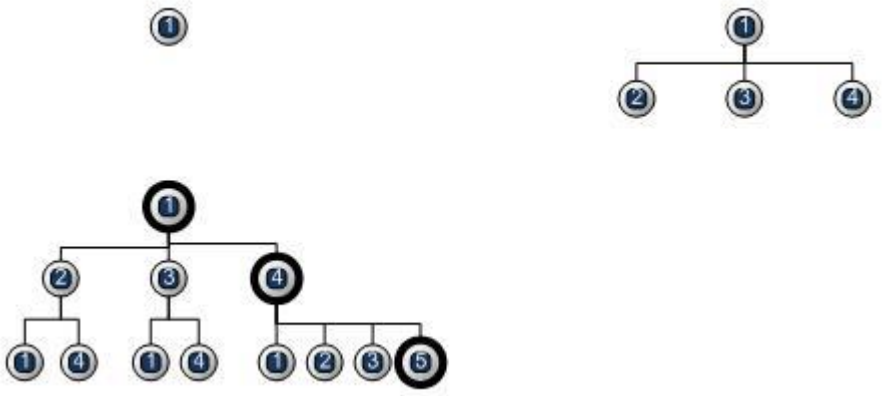
با توجه به گراف شکل بالا ، مسیر 5-4-3-1-2-1 ما را از گره 1 به گره شماره 5 هدایت می کند. با وجود اینکه محدود کردن عمق درخت مشکل حلقه بینهایت را رفع کرد، اما هنوز مشکل حالت تکراری در روش جستجوی عمقی محدود شده وجود دارد. برای حل همزمان مشکل حلقه بینهایت و حالت تکراری، می توان از روش جستجوی سطحی استفاده کرد که در بخش بعد به شرح آن پرداخته ایم

جستجوی سطحی

همانطور که در بخش جستجوی عمقی محدود شده بررسی کردیم، این روش جستجو مشکل حلقه بینهایت و یال های تکراری را حل کرد. البته این بدان معنا نیست که مشکل حالت های تکراری در جستجوی عمقی محدود شده به طور کامل رفع شده است. بلکه هیچیک از روش های جستجو در مقابل حالت های تکراری مصون نیستند. این وظیفه طراح الگوریتم است که با اتخاذ روشی از بروز حالت های تکراری جلوگیری کند. یکبار دیگر گراف زیر را در نظر بگیرید :



می خواهیم مسیری از گره 1 به گره 5 پیدا کنیم. در صورتی که از جستجوی سطحی برای پیمایش فضای جستجو بخواهیم استفاده کنیم، بدین روش عمل می کنیم : در هر سطح از درخت جستجو ، به ازای هر گره در آن سطح همه فرزندان گره جاری تولید شده و در سطح بعدی درخت قرار می گیرند. با این تعریف حال می توانیم نحوه پیمایش درخت جستجوی شکل 2 را نشان دهیم :

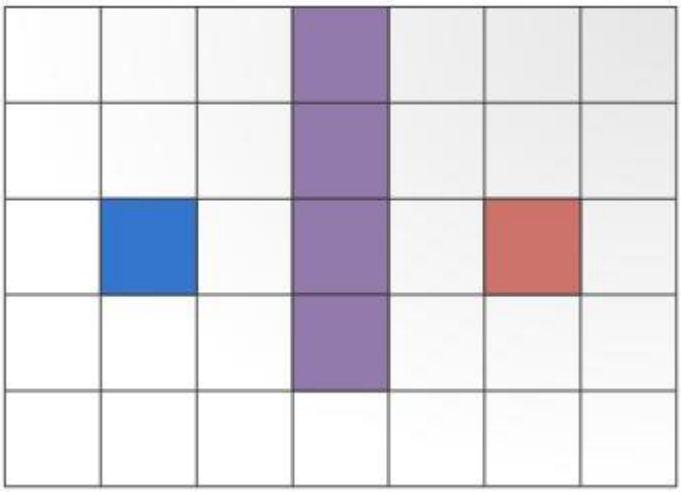


در ابتدا همه فرزندان گره 1 تولید شده و به سطح بعدی درخت اضافه می گردند. در سطح بعدی درخت نیز به ازای هر گره فرزندان آن گره تولید و به درخت جستجو اضافه می شوند. همانطور که از شکل پیداست، در سطح 3 از درخت مسیری از گره 1 به گره 5 پیدا می شود. همچنین مشکل یال های تکراری روش های قبلی نیز در این درخت وجود ندارد. یکی از بزرگترین مشکلات روش جستجوی سطحی مرتبه مکانی یا میران حافظه مصرفی این روش جستجو است. در روش های جستجو عمقی و عمقی محدود شده در هر لحظه تنها گره های مربوط به شاخه فعلی در حافظه وجود دارد. این در حالی است که روش جستجوی سطحی همه گره های بالاتر از سطح فعلی از فضای جستجو را در حافظه نگه خواهد داشت. می توان روش جستجوی عمقی و سطحی را باهم ترکیب کنیم تا با ترکیب این دو ، معایب موجود در هریک از روش ها را رفع کرده و مزایای آن ها را باهم ترکیب کنیم. این روش جستجو راجستجوی عمقی تکرار شونده می نامیم.

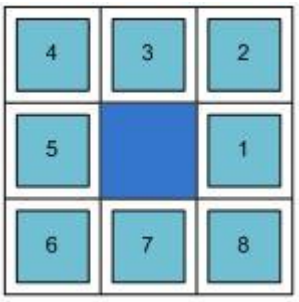
جستجوی هزینه یکنواخت

در هیچ یک از روش های جستجوی عمقی ، سطحی ، عمقی محدود شده و عمقی تکرار شونده هزینه ای که تا به حال از ریشه تا گره فعلی صرف شده است ، در نظر گرفته نمی شود. همین امر موجب می گردد تا در مسائل بهینه سازی نتوان از این روش ها برای جستجوی فضای جستجو استفاده کرد. چرا که هیچ استراتژی برای گسترش گره های بعدی وجود ندارد. از دیگر روش های جستجوی ناآگاهانه که بر روی درخت های وزن دار می توانیم از آن استفاده کنیم، روش جستجوی هزینه یکنواخت می باشد.

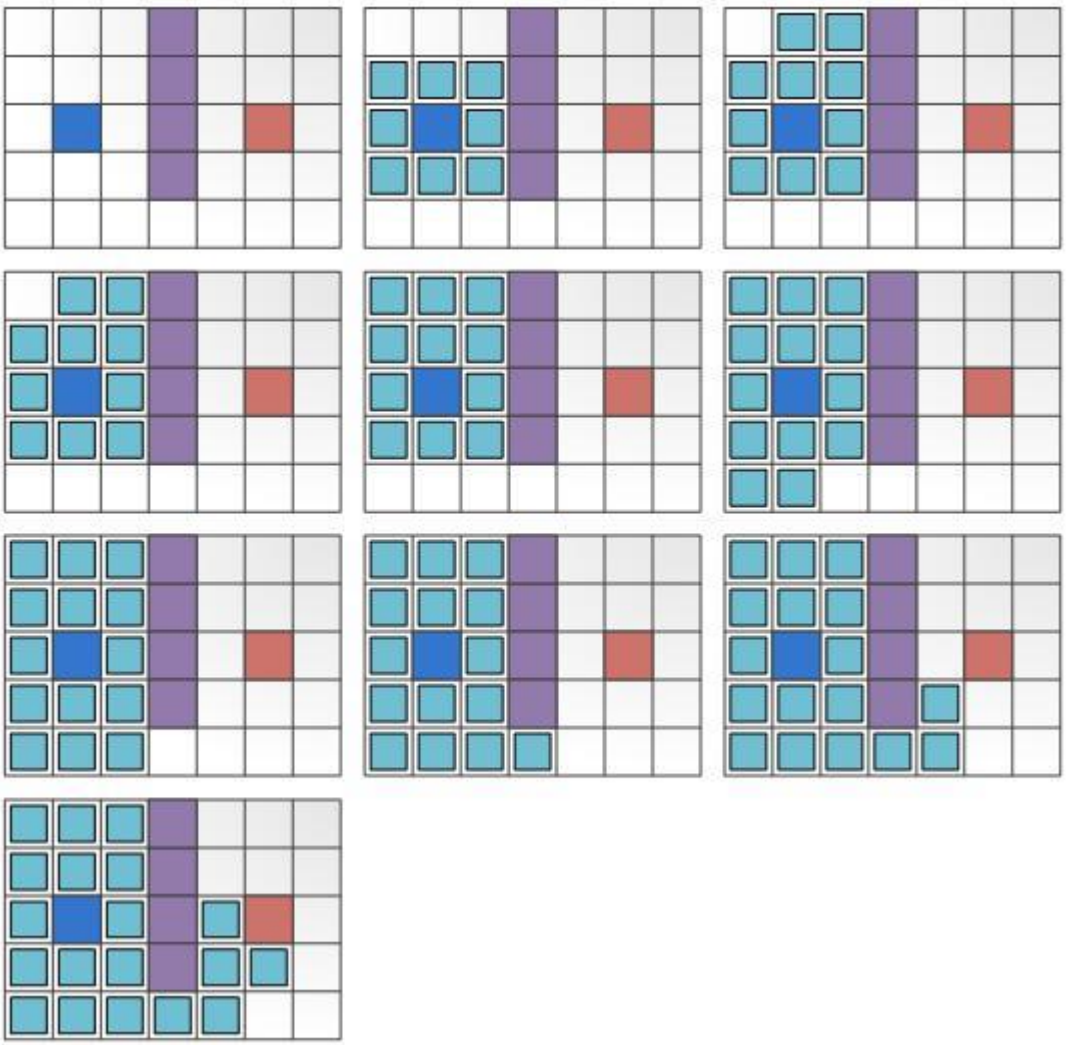
جستجوی هزینه یکنواخت را از آن جهت که در هر لحظه کم هزینه ترین گره را گسترش می دهد ، نه می توان در رده روش های جستجوی عمقی دانست ، نه می توان آن را یک روش جستجوی سطحی تلقی کرد. در این روش جستجو هر گره هزینه مصرف شده از ریشه تا گره فعلی را در خود نگه می دارد. هنگامی که می خواهیم فرزندان گره گسترش نیافته ای را تولید کنیم، گره ای از درخت برای گسترش انتخاب می شود که کم هزینه ترین گره در میان گره های گسترش نیافته باشد. به عنوان مثال شکل زیر را در نظر بگیرید :



مسئله یافتن مسیری از مستطیل آبی رنگ به مستطیل قرمز رنگ می باشد. همچنین مستطیل های بنفش نشان دهنده دیوار هستند که نمی توان بر روی آن ها حرکت کرد. در این مسئله به ازای هر خانه 8 خانه همسایه وجود دارد که در شکل زیر نشان داده شده اند (اعداد ترتیب گسترش گره ها را نشان می دهند) :

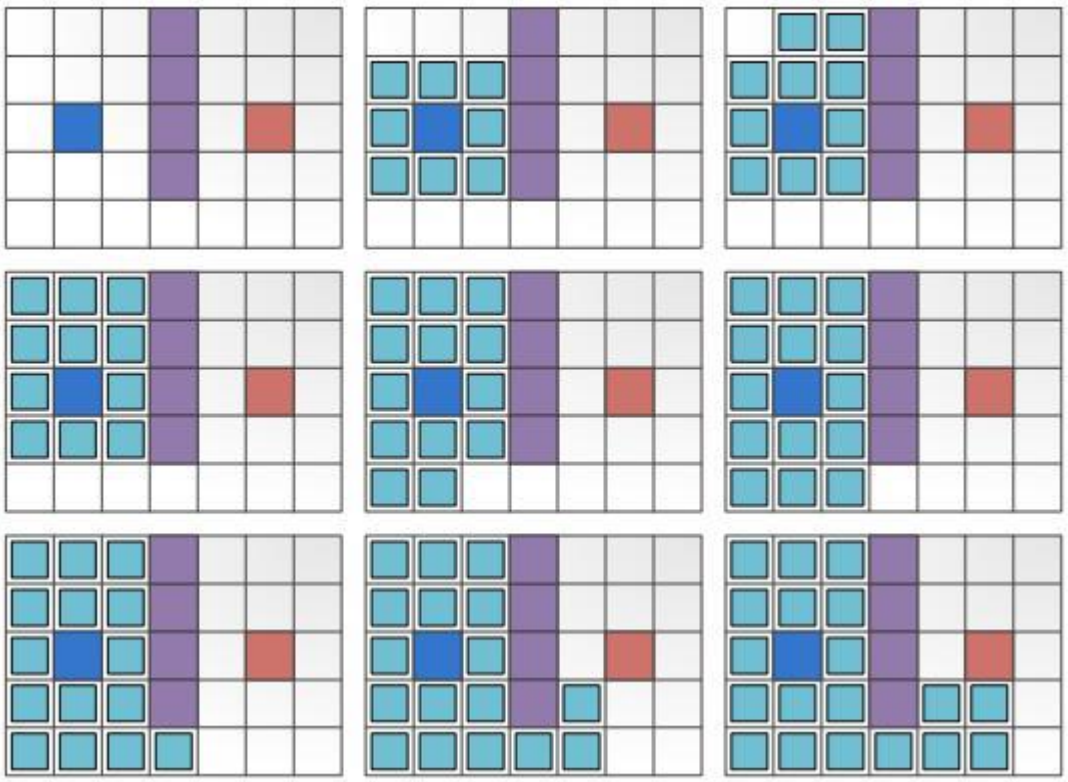


این بدان معناست که هنگام تولید فرزندان یک گره ، 8 خانه همسایه برای گسترش وجود خواهد داشت. از طرف دیگر حرکات قطری مجاز بوده و از یک خانه می توان چندبار عبور کرد. قبل از بررسی روش جستجوی هزینه یکنواخت با روش های جستجوی عمقی ، سطحی و عمقی محدود شده به حل این مساله می پردازیم. برای روش جستجوی عمقی بسته به اینکه ترتیب گسترش گره ها به چه نحوی باشد و همچنین مبدا و مقصد در کدام قسمت نقشه قرار گیرند، امکان قرار گرفتن در حلقه بینهایت وجود خواهد داشت. بنابراین از این روش استفاده نمی کنیم. در صورتی که از جستجوی سطحی برای حل مساله استفاده کنیم ، مسیر پیدا شده برای بدین صورت خواهد بود :

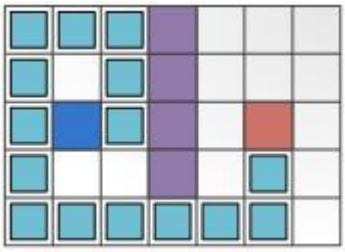


های اضافی بسیاری را نیز برای یافتن مسیر گسترش می دهد. که این امر در مورد مسائل پیچیده تر خوشایند نخواهد بود. حال فرض کنید از روش جستجوی عمقی محدود شده به عمق 20 برای یافتن مسیر استفاده کنیم.

شکل زیر نحوه گسترش گره ها را نشان می دهد :



بنابراین مسیر پیدا شده از مبدا به مقصد با روش جستجوی عمقی محدوده شده به صورت زیر خواهد بود :

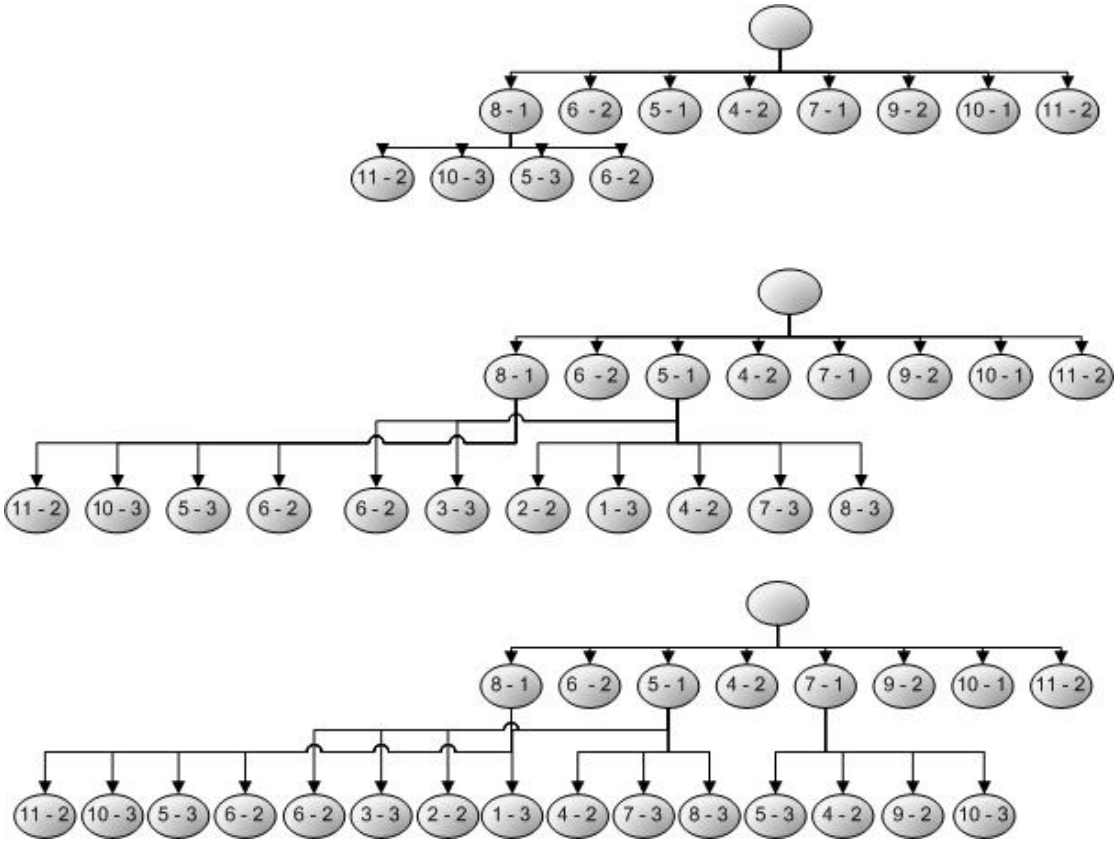


در مورد جستجوی عمقی محدود شده نیز با دو مشکل مواجه هستیم ، مشکل اول تعیین عمق محدود شده و مشکل دوم عدم بهینگی مسیر پیدا شده از مبدا به مقصد می باشد. در هیچیک از روش های جستجو فوق هزینه از ریشه تا گره فعلی در نظر گرفته نشده است. حال فرض کنید هزینه انتقال از یک خانه به خانه دیگر در جهت افقی

و عمودی 1 و در جهت قطری 1.5 باشد. حال با این تعریف می توانیم از روش جستجوی هزینه یکنواخت برای یافتن مسیر استفاده کنیم. همانطور که در ابتدای بخش بیان کردیم ، در روش جستجوی هزینه یکنواخت در هر لحظه گره ای از درخت گسترش می یابد که کمترین هزینه را در میان دیگر گره ها داشته باشد. قبل از نمایش نحوه تشکیل درخت جستجو به هر خانه شماره ای را انتساب می دهیم.

1	2	3		27	28	29
4	5	6		26	25	24
7		8		22		23
9	10	11		21	20	19
12	13	14	15	16	17	18

درختی که از جستجو به روش هزینه یکنواخت در هر مرحله به دست می آید به شکل زیر خواهد بود :



به دلیل بزرگ بودن درخت جستجو با استفاده از روش جستجوی هزینه یکنواخت در این مسئله ، تنها بخش کوچکی از آن را در اینجا نشان دادیم. بقیه گره ها نیز به همین ترتیب گسترش یافته تا اینکه در نهایت به گره هدف می رسیم. پس از ایجاد درخت خواهیم دید که مسیر بهینه 21-15-11 به عنوان مسیر حرکت از مبدا به مقصد انتخاب می شود.

نکته قابل توجه در مورد جستجوی هزینه یکنواخت این است که این روش جستجو در صورتی جواب بهینه را پیدا می کند که هزینه های هر گام به درستی انتخاب شوند. مشکلی که همچنان در این روش باقی مانده ، گسترش گره های اضافی است که این امر موجب می شود پیدا کردن جواب برای مسئله زیاد سریع نباشد. به جای جستجوی

هزینه یکنواخت می توان از جستجوی حریصانه نیز برای حل این مساله استفاده کرد که در بخش بعد به بررسی آن خواهیم پرداخت

روش های جستجوی هیوریستیک

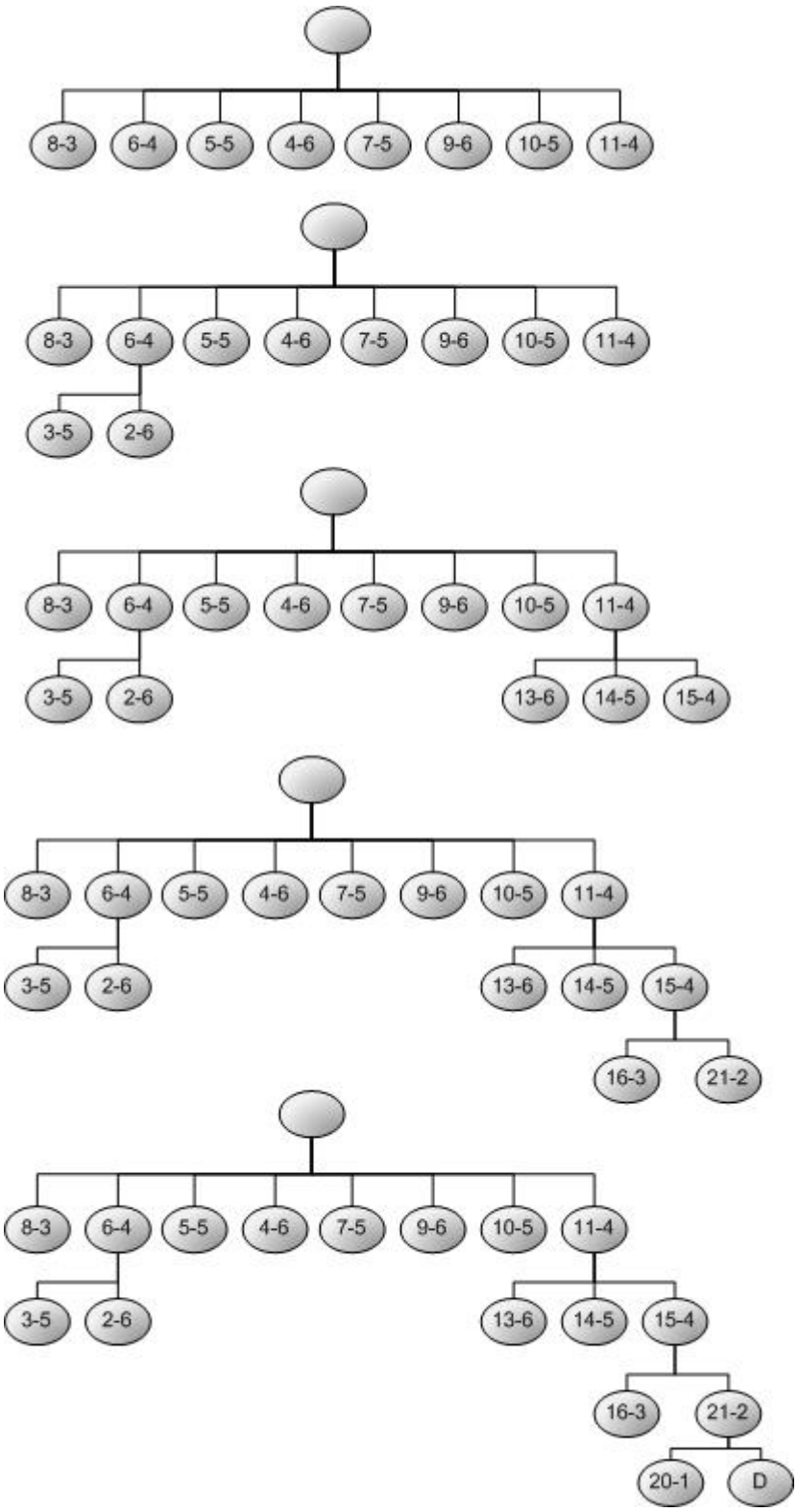
در روش های جستجوی عمقی، سطحی ، عمقی محدود شده ، عمقی تکرار شونده و هزینه یکتنواخت از هیچ تابعی برای تخمین میزان بهینگی هر گره استفاده نکردیم. تنها در جستجوی هزینه یکنواخت هزینه هر گام را برای سنجش میزان بهینگی گره ها به کار بردیم. اما در این روش جستجو نیز از اطلاعات مسئله برای تخمین میزان بهینگی هر گره بهره نبردیم. تابعی که عمل تخمین بهینگی هر گره را انجام می دهد ، تابع هیوریستیک می نامیم

جستجوی حریصانه

مسئله مسیریابی مطرح شده در بخش جستجوی هزینه یکنواخت را در نظر بگیرید. می توانیم به جای اینکه هزینه پیموده شده تا گره فعلی را به عنوان سنجه ای برای میزان بهینگی گره در نظر بگیریم، فاصله گره فعلی تا مقصد را به عنوان تخمینی برای سنجش میزان بهینگی هر گره انتخاب کنیم. به عبارت دیگر هنگام گسترش ، گره ای از درخت را گسترش دهیم که کمترین فاصله را تا مقصد دارد. در اینجا فرض می کنیم از هر خانه تنها یکبار می توانیم عبور کنیم. بنابراین تابع هیوریستیک به شکل زیر تعریف خواهد شد :

$$H(\text{گره}) = | X_{\text{گره}} - X_{\text{مقصد}} | + | Y_{\text{گره}} - Y_{\text{مقصد}} |$$

در این زیر مسئله گره ای برای گسترش مطلوتبر است که مقدار H آن کمتر باشد. درختی که با استفاده از این تابع هیوریستیک به دست می آید به شکل خواهد بود :


8-3
6-4
5-5
4-6
7-5
9-6
10-5
11-4
3-5
2-6
13-6
14-5
15-4
16-3
21-2
20-1
D

همانطور که در ابتدای مساله فرض کردیم ، از هر خانه تنها یکبار می توان عبور کرد. این بدان معنی است که هر گره تنها یکبار می تواند در درخت می تواند گسترش یابد. علت گسترش نیافتن گره 8 در سطح 1 از درخت نیز به همین دلیل است. چرا که همه فرزندان این گره قبلا در درخت (سطح 1) گسترش یافته اند.

درخت تشکیل شده از جستجوی حریصانه را با درخت تشکیل شده از جستجوی هزینه یکنواخت مقایسه کنید. می بینیم که جستجوی حزیصانه با گسترش گره های کمتر ، سریعتر به جواب مساله دست پیدا می کند. با این حال هنگام استفاده از جستجوی حریصانه باید دقت کنیم چرا که جستجوی حریصانه با دو مشکل بزرگ همراه است. مشکل اول اینکه نمی توان با استفاده از جستجوی حریصانه مطمئن بود که همیشه به جواب بهینه دست پیدا خواهیم کرد. همچنین جستجوی حریصانه ممکن است در یک حلقه بینهایت به دام افتد. به عنوان مثال در مسئله مسیریابی فرض کنید از هر خانه می توان چندین بار عبور کرد و با این فرض درخت جسنجو را تشکیل دهید. مشاهده خواهید کرد که گره شماره 8 در یک حلقه بینهایت گسترش یافته و در واقع هیچگاه به جواب مسئله دست پیدا نخواهیم کرد. حال می توان علت نامگذاری این روش به جستجوی حریصانه را حدس زد.

دیدیم که جستجوی هزینه یکنواخت در صورتی که هزینه گام ها به درستی انتخاب شود، موجب یافتن جواب بهینه مسئله خواهد شد. در مقابل این روش با مشکل کند بودن عمل جستجو همراه است. در مقابل جستجوی حریصانه در یافتن جواب مسئله بسیار سریع بوده اما با مشکل حلقه بینهایت و عدم بهینگی جواب مواجه است (لازم به ذکر است که بهینگی جستجوی حریصانه در برخی مسائل همانند الگوریتم های پریم و کروسکال اثبات شده است. اما در حالت کلی نمی توان ادعا کرد همیشه جستجوی حریصانه منجر به یافتن جواب بهینه خواهد شد). حال این سوال مطرح می شود که چگونه می توان این دو روش را باهم ترکیب کرده و روش جستجویی را طراحی کنیم که هم سریع بوده و هم جواب بهینه مسئله را بتواند پیدا کند؟ جواب مسئله در روش جستجوی A* است