دانشکده مهندســی

پایان‌نامه کارشناسی ارشد در رشته مهندسی کامپیوتر (نرم افزار)

الگوریتم‌های جستجوی مکاشفه‌ای بلادرنگ

توسط:

روح‌الله تقی‌زاده

استاد راهنما:

دکتر کورش زیارتی

شهریور ۱۳۸۸

به نام خدا

الگوریتم‌های جستجوی مکاشفه‌ای بلادرنگ

به وسیله ی:

روح‌الله تقی‌زاده

پایان نامه

ارائه شده به تحصیلات تکمیلی دانشگاه به عنوان بخشی

از فعالیت‌های تحصیلی لازم برای اخذ درجه کارشناسی ارشد

در رشته:

مهندسی کامپیوتر - نرم افزار

از دانشگاه شیراز

شیراز

جمهوری اسلامی ایران

ارزیابی شده توسط کمیته پایان نامه، با درجه: عالی

دکتر کورش زیارتی، استادیار بخش مهندسی کامپیوتر (رئیس کمیته) .....

دکتر منصور ذوالقدری جهرمی، دانشیار بخش مهندسی کامپیوتر .....

دکتر محمد هادی صدرالدینی ، دانشیار بخش مهندسی کامپیوتر .....

شهریور ماه ۱۳۸۸

تقدیم

به قلب پاک مادرم تقدیم میکنم، که روح آب است

و به چشمهای مهربان پدرم، که آیینه آفتاب است.

سپاسگزاری

پیش از هر چیز از خداوند مهربان سپاسگزارم که همیشه از روی لطف و کرم خود با من رفتار کرده است و سپس از خانواده‌ام که همیشه در تمام مراحل زندگی پشتیبان من بوده‌اند سپاسگزاری می‌کنم.

از استاد ارجمند جناب آقای دکتر کورش زیارتی که در تمام مراحل این پایان‌نامه از هیچگونه کمکی دریغ نکردند کمال تشکر و قدردانی را دارم.

در پایان از تمام دانشجویان کارشناسی ارشد و دکتری بخش مهندسی و علوم کامپیوتر که در طول ۳ سال دوره کارشناسی ارشد، در تمامی لحظات، همراه و همدل من بوده‌اند سپاسگزاری می‌نمایم.

فرصت شمار صحبت، کز این دو راهه منزل

چون بگذریم دیگر، نتوان به هم رسیدن

به امید حق

چکیده

الگوریتم‌های جستجوی مکاشفه‌ای بلادرنگ

به کوشش:

روح‌الله تقی‌زاده

جستجوهای مکاشفه‌ای یکی از شناخته شده ترین روشهای هوش مصنوعی برای حل مسئله یافتن کوتاهترین مسیر (مسیری با کمترین هزینه) در یک گراف هستند. در این پایان نامه، الگوریتم‌های جستجوی مکاشفه‌ای جدیدی برای سرعت بخشیدن و افزایش توانایی الگوریتم‌های موجود در جهت حل مسائل بزرگتر ارائه شده است.

در این پایان‌نامه یک جستجوی اول بهترین مانند *WA که شناخته شده است گسترش داده می‌شود تا برای یافتن جوابهایی با هزینه یکسان، حافظه کمتری مصرف کند و گره های کمتری تولید کند. این کار جهت افزایش توانایی الگوریتم برای حل مسائل بزرگتر انجام می شود. در این راه، از ایده‌های جدیدی برای هرس کردن بعضی از شاخه های درخت استفاده می‌شود تا تعداد گره های تولید شده، کاهش و سرعت جستجو افزایش یابد.

کارهای انجام شده در این پایان نامه به دو بخش تقسیم می‌شود. در بخش اول، راهکاری برای بهبود الگوریتم *WA ارائه شد. این راهکار، ابتدا با یک وزن زیاد یک جواب سریع بدست می‌آورد و سپس با روشهایی، سعی در بهبود جواب بدست آمده دارد. در بخش دوم، جستجوی مکاشفه‌ای با جستجوی Tabu ترکیب شده و الگوریتم‌های *Tabu A و *Tabu IDA ارائه می‌گردند. عملکرد راهکارها و الگوریتم‌های ارائه شده با عملکرد الگوریتم *WA مقایسه می‌شود. نتایج نشان می‌دهند که الگوریتم‌های ارائه شده نسبت به الگوریتم *A دارای عملکرد بهتری می‌باشند.

فهرست مطالب

فهرست شکل‌ها

فصل اول

مقدمه

۱- مقدمه

۱-۱- جستجوی مکاشفه‌ای

یکی از چهارچوب‌های حل مسئله که بسیار از آن استفاده می‌شود جستجوی مکاشفه‌ای[۱] برای یافتن یک مسیر با حداقل هزینه در گراف می‌باشد. الگوریتم‌های زیادی برای حل این مسئله آورده شده است و تحقیقات بر روی این حوزه همچنان ادامه دارد. یکی از الگوریتم‌های پایه‌ای A^* می‌باشد و الگوریتم‌های یک‌سویه[۲] و دوسویه[۳] زیادی مانند IDA^*[۴]، *Bidirectional* A^*، *Perimeter Search*، ... از آن منشعب شده است [۱,۲,۳,۴,۵,۶].

با پیشرفت تکنولوژی و روی کار آمدن سخت افزارهای سریع، نیاز به الگوریتمها و نرم‌افزارهایی که بتوانند سخت‌افزارهای سریع را همراهی کنند بیش از پیش معلوم می‌شود. یافتن یک مسیر با حداقل هزینه در یک گراف بزرگ کاری بسیار پیچیده و زمان‌بر می‌باشد و به علت فراوانی کاربردهای این مسئله، همواره مورد توجه محققان بوده است. الگوریتمهای موجودی که برای حل این مسئله ارائه شده‌اند، نظیر جستجوهای مکاشفه‌ای، امکان یافتن جواب بهینه را دارند اما زمان بدست آوردن این جواب بسیار زیاد است.

یافتن مسیر بهینه برای مسائل بزرگ و پیچیده می‌تواند زمان زیادی صرف کند و بدست آوردن یک جواب نزدیک به بهینه[۵] که در زمان کوتاه و بلادرنگ[۶] حاصل شود، می‌تواند بسیار مفید باشد. مسائل زیادی در واقعیت وجود دارند که فضای آنها به صورت یک گراف بوده و می‌توان آنها را با جستجوی مکاشفه‌ای حل کرد و از آنجا که اکثر مسائل واقعی مانند برقراری

[1] Heuristic Search
[2] Unidirectional
[3] Bidirectional
[4] Iterative Deepening A*
[5] Suboptimal
[6] Real-Time

ارتباطات مخابراتی راه دور، ارتباطات شبکه اینترنت، همترازی چند دنباله[۷]، بزرگترین زیر رشته مشترک[۸] و ... احتیاج به جواب سریع دارند، جستجوهای بلادرنگ می‌توانند بسیار مفید باشند.

تکنیک‌های مختلفی برای تغییر الگوریتم‌های جستجوی مکاشفه‌ای وجود دارد که اجازه برقراری تعادل[۹] را بین کیفیت جواب و زمان جستجو می‌دهد. بعضی از این تکنیک‌ها برای یافتن یک جواب خوب اما نه لزوماً بهینه در زمان کوتاه برنامه‌ریزی شده‌اند که می‌توان برای مثال از جستجوی مکاشفه‌ای وزن‌دار[۱۰] نام برد. فرض در این تکنیک‌ها این است که هنگامیکه اولین جواب به دست آمد الگوریتم متوقف شده و جواب را بر می‌گرداند. بعضی دیگر از تکنیک‌ها تا زمان یافتن جواب بهینه یا ارسال یک وقفه[۱۱] خارجی به کار خود ادامه داده و سعی در بهتر کردن جواب پیدا شده دارند. الگوریتمهایی نیز با ایده گرفتن از بازیهای دونفره[۱۲] ارائه شده‌اند که احتیاج به تصمیم گیری برای یک حرکت در زمان معین دارند.

پایه‌ای‌ترین الگوریتم جستجوی مکاشفه‌ای که در سال ۱۹۶۸ ارائه شده است A^* می‌باشد [۷]. این الگوریتم از یک تابع ارزیابی[۱۳] برای تخمین زدن فاصله گره فعلی تا گره هدف استفاده می‌کند. به دلیل اینکه A^* تمام گره‌های تولید شده برای رسیدن به هدف را نگهداری می‌کند فضای زیادی مصرف کرده و عملاً برای مسائل بزرگ کارآیی ندارد. اگر تابع ارزیابی مورد استفاده در این الگوریتم پذیرفتنی[۱۴] باشد(یعنی همیشه فاصله موقعیت فعلی تا هدف را کمتر از مقدار واقعی حدس بزند) اولین جواب بدست آمده بهینه می‌باشد.

الگوریتم‌های مختلفی بر پایه الگوریتم A^* ارائه شده‌اند. گروهی از آنها به منظور کاهش میزان حافظه مصرفی ارائه شده‌اند که می‌توان به جستجوی A^* عمیق‌شونده تکراری IDA^* و جستجوی اول بهترین بازگشتی $RBFS$ اشاره کرد [۱,۲] و گروهی برای کاهش تعداد گره‌های

---

[7] Multiple Sequence Alignment
[8] Longest Common Subsequence
[9] Tradeoff
[10] Weighted Heuristic Search
[11] Interrupt
[12] Two Player Game
[13] Evaluation Function
[14] Admissible

تولید شده بکار رفته‌اند که می‌توان از جستجوی A^* دوسویه *Bidirectional A*، جستجوی پیرامونی *Perimeter Search* و *BIDA نام برد [۴,۵,۶]. گروهی دیگر نیز برای سرعت بخشیدن به عملیات جستجوی در ازای نابهینه بودن جواب ارائه شده‌اند. یک ایده برای این مسئله وزن دادن به تابع ارزیابی پذیرفتنی و تبدیل آن به غیرپذیرفتنی و استفاده از جستجوی مکاشفه‌ای برای یافتن سریع جواب می‌باشد که جستجوی مکاشفه‌ای وزن‌دار نامیده می‌شود [۸,۹]. در این الگوریتم فرض بر آن است که هنگامیکه اولین جواب بدست آمد الگوریتم متوقف شده و جواب بدست آمده برگردانده می‌شود. تحلیل این الگوریتم بر توصیف تعادل بین زمان لازم برای بدست آوردن اولین جواب و کیفیت آن متمرکز می‌شود. به عنوان مثال، این نشان داده شده است که در این الگوریتم هزینه اولین جواب پیدا شده بیشتر از هزینه جواب بهینه ضربدر ضریب $1+\varepsilon$ نخواهد بود که ε به مقدار وزن ضرب شده در تابع ارزیابی بستگی دارد [۱۰,۱۱]. همچنین نتایج تجربی تعادل بین زمان جستجو و کیفیت جواب بدست آمده است [۱۲]. همانطور که گفته شد در این الگوریتم‌ها هنگامیکه اولین جواب بدست می‌آید الگوریتم متوقف می‌شود. بعدها امکان ادامه یافتن الگوریتم پس از یافتن اولین جواب مطرح شد [۱۳] و سپس این ایده به آن اضافه شد که می‌توان با افزایش زمان محاسبات به کیفیت جواب افزود که به نام *Anytime* معروف است [۱۴]. سپس گروه‌های مختلفی برای بکار بردن این ایده‌ها در جستجوهای مکاشفه‌ای اقدام کردند که منجر به ارائه الگوریتم‌های مختلفی نظیر *Anytime A** شد [۱۵,۱۶]. در سالهای اخیر(۲۰۰۷) نیز الگوریتم‌های *Anytime Weighted A** و *Anytime RBFS* ارائه گردیده است [۱۷].

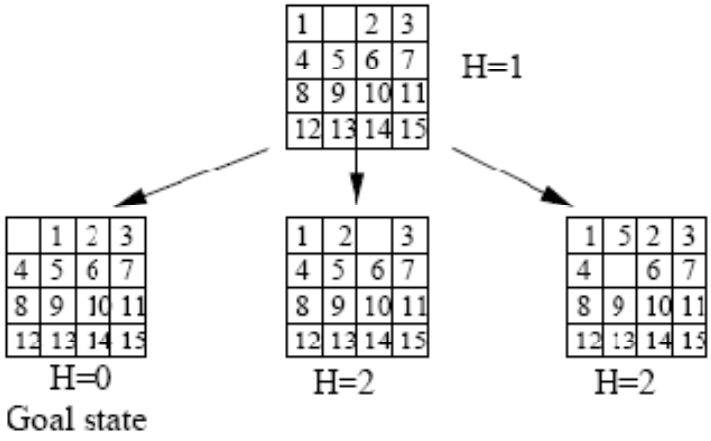
در این پایان‌نامه سعی بر آن است تا راهبردهای ارائه شده برای یافتن یک جواب سریع برای مسئله یافتن کوتاهترین مسیر بین دو گره از یک گراف مورد بررسی قرار گیرد و راهبردهای جدیدی برای حل این مسئله ارائه شود. از آنجا که یافتن جواب بهینه برای این مسئله بسیار وقتگیر می‌باشد و مسائل واقعی موجود نظیر ارتباطات شبکه اینترنت و مخابرات مجهز به سخت‌افزارهای سریع شده‌اند راهبردهایی که بتواند یک جواب خوب را در زمان کم ارائه دهد از اهمیت بسیار بالایی برخوردار است. در این راستا تحقیقات زیادی انجام و

الگوریتم‌هایی نظیر *WA* و *Anytime A* ارائه شده است اما هنوز نیازهای مسائل دنیای واقعی برآورده نشده و تحقیقات زیادی مورد نیاز است.

۱-۲- پازل کاشی‌های لغزنده[15]

پازل کاشی‌های لغزنده یک مسئله جستجوی مشهور است [۱,۲]. یکی از پرطرفدارترین نسخه‌های آن شامل یک آرایه ۴*x*۴ با ۱۵ کاشی مربع شکل با شماره‌های ۱ تا ۱۵ و یک جای خالی می‌باشد. هدف این پازل حرکت دادن کاشی‌ها به سمت موقعیت هدف می‌باشد تا در نهایت کاشی‌ها با ترتیب خاصی قرار گیرند. در هر حرکت هر کاشی که در کنار جای خالی قرار گرفته است میتواند به مکان جای خالی منتقل شود. شکل ۱-۱ پازل ۱۵ را نمایش می‌دهد. یک تابع مکاشفه‌ای عمومی برای این پازل فاصله منهتن[16] می‌باشد که مجموع فاصله افقی و عمودی هر کاشی از مکانش در موقعیت هدف می‌باشد. این مقدار یک حد پایین برای هزینه واقعی جواب می‌باشد زیرا هر کاشی حداقل به اندازه فاصله منهتن خود باید حرکت کند و در هر حرکت تنها یک کاشی جابجا می‌شود.

[15] Sliding Tile Puzzle
[16] Manhattan Distance



شکل ۱-۱- پازل ۱۵

یکی از اولین تحقیقاتی که در حوزه جستجوی مکاشفه‌ای روی پازل کاشی‌های لغزنده انجام شد یافتن جواب بهینه برای آن بود. پازل ۸[۱۷] در اولین تلاشها در این حوزه بصورت بهینه حل گشت. این پازل روی سیستمهای کامپیوتر فعلی میتواند بصورت بهینه توسط الگوریتم A^* و حتی الگوریتم جستجوی سطحی بطور بهینه حل شود اما پازل ۱۵ [۱۸] به علت فضای زیادی که الگوریتم A^* مصرف می‌کند توسط این الگوریتم قابل حل نیست. این مشکل تا سال ۱۹۸۵ که جستجوی A^* عمیق‌شونده تکراری[۱۹] ابداع شد وجود داشت[۱]. الگوریتم IDA^* هر نمونه پازل ۱۵ را با استفاده از تابع مکاشفه‌ای منهتن بطور متوسط در حدود یک دقیقه حل می‌کند.

در ادامه خواهیم دید که الگوریتم‌های جستجوی مکاشفه‌ای مختلف احتیاج به یک تابع مکاشفه‌ای جهت تخمین مینیمم هزینه رسیدن از یک حالت به حالت هدف دارند.

[17] 8 Puzzle
[18] 15 Puzzle
[19] Iterative Deepening A* (IDA*)

فصل دوم

جستجوی مکاشفه‌ای بهینه

۲- جستجوی مکاشفه‌ای بهینه

۲-۱- جستجوی مکاشفه‌ای

جستجوی مکاشفه‌ای[۱] یک مکانیسم حل مسئله عمومی[۲] در حوزه هوش مصنوعی[۳] می‌باشد. این جستجو در مسائلی بکار می‌رود که فضای جستجوی آنها یک فضای گراف[۴] باشد. گره‌های گراف نمایانگر حالتهای[۵] مسئله و یالها نشانگر حرکتهای مجاز[۶] می‌باشند. یک نمونه مسئله[۷] شامل فضای مسئله و حالت ابتدایی[۸] یا شروع و مجموعه‌ای از حالات هدف[۹] است. در این گراف یک مسیر از حالت شروع به یکی از حالت‌های هدف را جواب[۱۰] گویند. هزینه یک جواب، مجموع هزینه یالهای آن می‌باشد. یک جواب را بهینه[۱۱] گویند اگر کم هزینه‌ترین مسیر از حالت شروع به یکی از حالت‌های هدف باشد در غیر اینصورت زیر بهینه[۱۲] است. برای پیاده‌سازی گراف باید از یک ساختمان داده[۱۳] استفاده کرد. زمانیکه یک ساختمان جدید برای نگهداری یک گره ایجاد می‌شود از اصطلاح تولید کردن[۱۴] یک گره استفاده می‌شود. گسترش دادن[۱۵] یک گره به تولید کردن تمام فرزندان آن اتلاق می‌شود.

---

[1] Heuristic Search
[2] General Problem-Solving
[3] Artificial Intelligence
[4] Problem Space Graph
[5] State
[6] Legal Moves
[7] Problem Instance
[8] Initial State
[9] Goal State
[10] Solution
[11] Optimal
[12] Suboptimal
[13] Data Structure
[14] Generating
[15] Expanding

در واقع فضای جستجو[16] بصورت یک درخت جستجو[17] می‌باشد که ریشه آن حالت ابتدایی مسئله است و جستجو از این حالت ابتدایی شروع می‌شود. الگوریتم‌های جستجوی مختلف در نحوه و ترتیب حرکت کردن در درخت جستجو متفاوتند. این تفاوت از تصمیم‌گیری در مورد اینکه کدام گره در مرحله بعد گسترش یابد، نشأت می‌گیرد. الگوریتم‌های مختلف بر اساس سه معیار زیر با یکدیگر مقایسه می‌شوند:

پیچیدگی زمانی[18]: نظر به اینکه تولید یک گره معمولا زمان ثابتی طول می‌کشد پیچیدگی زمانی رابطه مستقیمی با تعداد گره‌های تولید شده دارد.

پیچیدگی فضایی[19]: یک الگوریتم نسبت به حافظه‌ای که برای حل مسئله مصرف می‌کند سنجیده می‌شود.

کیفیت جواب[20]: یک الگوریتم می‌تواند جوابی با کیفیت‌های مختلف برگرداند. بطور مثال یک جواب می‌تواند بهینه باشد یعنی مسیری با کمترین هزینه از حالت ابتدایی تا حالت هدف باشد و یا زیر بهینه باشد که در اینصورت تضمینی برای بهینه بودن این جواب وجود ندارد. بعضی وقتها حدی برای کیفیت جواب قرار داده میشود و گفته می‌شود که مقدار جواب به اندازه یک ضریب ثابت بزرگتر از جواب بهینه نیست.

۲-۲- جستجوی اول بهترین

جستجوی اول بهترین[21] [۷] یک الگوریتم شناخته شده و عمومی در حوزه جستجوی مکاشفه‌ای می‌باشد. این الگوریتم از یک لیست باز[22] برای نگهداری گره‌های تولید شده‌ای که

[16] Search Space
[17] Search Tree
[18] Time Complexity
[19] Space Complexity
[20] The Quality of the Solution
[21] Best-First Search (BFS)
[22] Open List

هنوز گسترش نیافته‌اند استفاده کرده و در هر مرحله امیدبخش‌ترین گره (بهترین گره) این لیست را برای گسترش انتخاب می‌کند. برای تعیین بهترین گره از یک تابع ارزیابی[۲۳] f استفاده می‌شود. گرهی که کمترین مقدار تابع ارزیابی را داشته باشد امیدبخش‌ترین گره برای رسیدن به هدف است و برای گسترش انتخاب می‌شود. زمانیکه یک گره گسترش می‌یابد از لیست باز حذف شده، تمام همسایه‌های آن تولید شده و به لیست باز اضافه می‌شوند. معمولا زمانی جستجو خاتمه می‌یابد که یک گره هدف برای گسترش انتخاب گردد (این گره و مسیر رسیدن به این گره به عنوان جواب مسئله برگردانده می‌شود) و یا لیست باز خالی شود (مسئله جواب ندارد). جستجوی اول بهترین زیر شاخه‌های متنوعی شامل جستجوی اول سطح[۲۴] (جستجوی سطحی)، الگوریتم کوتاه‌ترین مسیر از یک مبدا دیکسترا[۲۵] و الگوریتم A^* [۷] دارد که این الگوریتم‌ها تنها در نحوه محاسبه تابع هزینه[۲۶] $f(n)$ با یکدیگر تفاوت دارند. اگر هزینه یک گره برابر عمق آن گره در درخت جستجو محاسبه گردد جستجوی اول بهترین همان جستجوی اول سطح بوده و تمام گره‌های یک سطح را قبل از گره‌های سطح بعد گسترش می‌دهد. در گرافهایی که یالهای آن هزینه‌های متفاوتی دارند اگر هزینه گره n برابر $g(n)$ (مجموع هزینه یالهای مسیر از گره ابتدایی تا گره n) محاسبه شود جستجوی اول بهترین همان جستجوی دیکسترا خواهد بود. در جستجوی A^* تابع هزینه از رابطه $f(n)=g(n)+h(n)$ محاسبه می‌شود که $h(n)$ تخمینی از هزینه مسیر باقیمانده از گره n تا گره هدف می‌باشد [۷]. در بخش‌های بعد توضیح مفصل‌تری بر الگوریتم A^* ارائه می‌شود.

۲-۳- الگوریتم A^*

الگوریتم A^* یک الگوریتم جستجوی اول بهترین است که که از لیست گره‌های کاندیدا، بهترین گره را برای گسترش انتخاب می‌کند. این لیست که باز نامیده می‌شود تمام گره‌های تولید شده

[23] Evaluation Function
[24] Breadth-First Search
[25] Dijkstra's Single-Source Shortest-Path Algorithm
[26] Cost Function

که تا بحال گسترش نیافته‌اند را در خود نگهداری می‌کند. بهتر بودن یک گره از این لیست توسط یک تابع تخمین[۲۷] تعیین می‌شود. این تابع هزینه کوتاهترین مسیر[۲۸] از گره شروع[۲۹] تا گره هدف[۳۰] که از این گره عبور می‌کند را تخمین می‌زند. این تخمین که با f نمایش داده می‌شود مجموع هزینه کوتاهترین مسیر پیدا شده تا بحال از گره شروع تا این گره (مقدار g[۳۱] این گره) و تخمینی از مسیر باقیمانده از این گره تا گره هدف (مقدار h[۳۲] این گره) می‌باشد. به علت اینکه هنوز کوتاهترین مسیر از یک گره تا گره هدف شناخته نشده است مقدار h یک گره توسط یک تابع مکاشفه‌ای[۳۳] محاسبه می‌گردد. اگر این تابع پذیرفتنی[۳۴] باشد (یعنی مقدار h هیچ گرهی را از مقدار حقیقی بالاتر تخمین[۳۵] نزند) در آنصورت A^* یک الگوریتم کامل[۳۶] است و جواب بهینه[۳۷] برمی‌گرداند و نسبت به تمام الگوریتم‌های جستجوی اول بهترین، از لحاظ کارآیی[۳۸] نیز بهینه است [۱۰,۱۸].

۲-۴- الگوریتم‌های فضای خطی[۳۹]

اگرچه جواب برگردانده شده توسط الگوریتم A^* بهینه (در صورت پذیرفتنی بودن تابع مکاشفه‌ای) می‌باشد [۱۸] اما از معایب آن پیچیدگی فضا و زمان بالای آن است [۱]. اصلی‌ترین اشکال این الگوریتم، احتیاج به حافظه آن می‌باشد. این الگوریتم تمام گره‌های لیست باز را برای تضمین بهینگی جواب و تمام گره‌های لیست بسته را برای برگرداندن مسیر جواب هنگامیکه به یک هدف می‌رسد در حافظه ذخیره می‌کند. پیچیدگی فضای A^* مانند پیچیدگی زمان آن به

[27] Estimate Function
[28] Shortest Path
[29] Start State
[30] Goal State
[31] g-value
[32] h-value
[33] Heuristic Function
[34] Admissible
[35] Overestimate
[36] Complete
[37] Optimal
[38] Efficiency
[39] Linear Space Algorithms