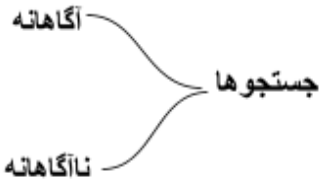
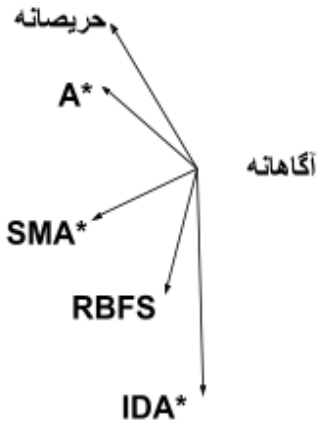
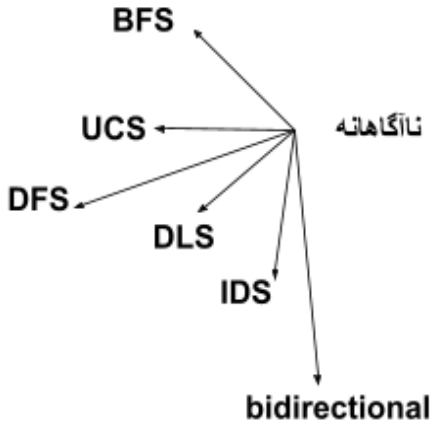
خلاصه و نکات


جستجوها
آگاهانه
ناآگاهانه


آگاهانه
حریصانه
A*
SMA*
RBFS
IDA*


ناآگاهانه
BFS
UCS
DFS
DLS
IDS
bidirectional

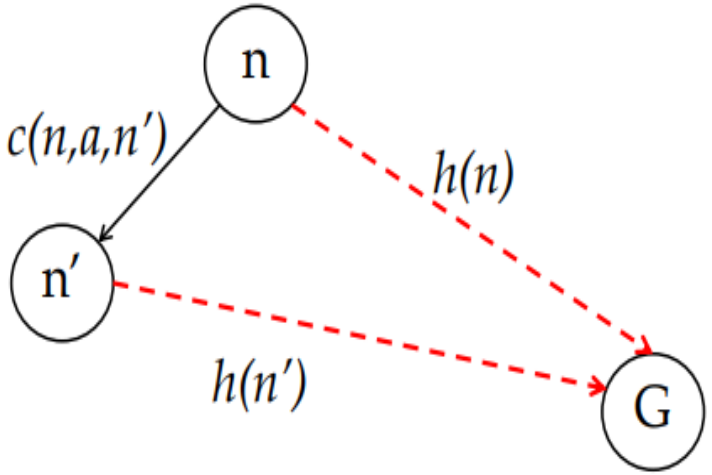
- **قابل قبول بودن و سازگار بودن هیوریستیک ها :**
-

قابل قبول بودن : هیوریستیک(n)hقابل قبول است اگر هزینه مسیر هر گره تا هدف را بیشتر از مقدار واقعی تخمین نزند.

- سازگاری Consistent
  - هیوریستیک $h(n)$ سازگار است اگر برای هر گره n و هر پسین آن مانند n' که با انجام عمل a به آن برسیم داشته باشیم:

$$h(n) \leq c(n,a,n') + h(n')$$

که در آن $c(n,a,n')$ برابر با هزینه مرحله‌ای رفتن از n به n' با انجام عمل a است.



یکنوا سازی

- $f(n) \leq f(next\ node\ after\ n)$

جستجوی اول بهترین بازگشتی– RBFS

ساختاری شبیه به جستجوی عمقی بازگشتی دارد اما به‌جای این‌که دائماً مسیر فعلی را به سمت پایین ادامه دهد، مقدار f بهترین مسیر جانشین از طریق اجداد گره فعلی را نگه می‌دارد. اگر f گره فعلی از این حد تجاوز کند، الگوریتم به عقب برمی‌گردد تا مسیر جانشین را انتخاب نماید. در برگشت به عقب این الگوریتم مقدار f مربوط به بهترین برگ در زیردرخت فراموش‌شده را به یاد می‌آورد و می‌تواند تصمیم بگیرد آیا این زیردرخت باید بعداً دوباره ایجاد شود یا خیر.

نکات SMA* :

الگوریتم SMA*

- ایده: استفاده از تمامی حافظه موجود
  - یعنی، گسترش بهترین گره‌ی برگ تا زمانی که حافظه موجود پر شود.
- SMA* دقیقا مثل A* عمل می‌کند، یعنی تا زمانی که حافظه پر شود، بهترین گره (گره‌ای با کم‌ترین مقدار f) را گسترش می‌دهد.
- در صورت پر شدن حافظه همیشه بدترین گره برگ (گرهی با بیشترین مقدار f) را از حافظه حذف می‌کند
- مشابه با RBFS، اطلاعات گره‌ی حذف شده را در گره پدرش ذخیره می‌کند تا در صورتی که شاخه‌های دیگر به خوبی این شاخه نبودند دوباره به این شاخه برگردد.

الگوریتم SMA*

- ممکن است f-cost تمام گره‌های برگ با هم برابر باشند در این‌صورت ممکن است گره‌ای که برای بسط دادن انتخاب شده، (به علت پر بودن حافظه) برای حذف نیز انتخاب شود! برای پرهیز از این مشکل، SMA* همیشه بهترین گره برگی که از همه جدیدتر (عمیق‌تر) است را برای بسط دادن انتخاب می‌کند و همیشه بدترین گره‌ای که از همه قدیمی‌تر (کم عمق‌تر) است را برای حذف انتخاب می‌کند.
- ممکن است به حالتی برسیم که فقط یک برگ (که هدف هم نیست) در درخت باشد و حافظه نیز پر باشد. در این‌صورت نمی‌توان آن گره برگ را بسط داد (چون حافظه خالی نداریم) و اگر آن برگ در مسیر بهینه باشد الگوریتم SMA* نمی‌تواند با مقدار حافظه موجود مسیر بهینه را پیدا کند. در این موارد، SMA* مقدار f-cost این برگ را بی‌نهایت می‌گذارد تا دیگر انتخاب نشود.