ساختمان داده
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی. پاییز ۱۴۰۲

۱ معرفی ساختار داده درخت اسپلی

درخت اسپلی که توسط دانیل اسلیتر و رابرت تارجان طراحی شده است در واقع یک BST است با این تفاوت اساسی که عمل جستجو در آن باعث تغییر ساختار درخت می‌شود. در BST معمولی (و خیلی از ساختار داده‌های بر پایه BST) عمل جستجو تغییری در ساختار درخت ایجاد نمی‌کند. از این رو درخت اسپلی یک درخت پویاست و مدام در حال تغییر است. در درخت اسپلی عنصر مورد جستجو با یک سری عمل دوران rotation به ریشه درخت آورده می‌شود. این ایده ساده به طرز اعجاب آوری باعث تسریع اجرای جستجوها می‌شود به نحوی که اگر دنباله عناصر مورد جستجو به اندازه کافی طولانی باشد عملکرد درخت اسپلی قابل مقایسه با درختان BST متوازن است و حتی در مواردی از آنها بهتر عمل می‌کند. این در حالی است که درخت اسپلی بر خلاف ساختارهای متوازن (مانند AVL) اطلاعاتی اضافه بر درخت ذخیره نمی‌شود.

۱.۱ عمل اسپلی

یک عمل اساسی در Splay Tree اسپلی کردن یک راس است. فرض کنید x یکی از عناصر BST باشد. با استفاده از دنباله‌ای از دورانها rotations عنصر x به ریشه درخت آورده می‌شود. دقت کنید دوران ها به طرز خاصی انجام می‌شود (هر گونه دورانی قابل قبول نیست.)

بطور کلی اسپلی کردن یک عنصر شامل اجرای ترکیبی از سه نوع عمل ابتدایی است که در زیر معرفی می‌شوند.

- حالت اول: زیگ zig

  این حالتی است که x پدربزرگ ندارد. در این حالت x با یک دوران به ریشه می‌رود.

(a)

- حالت دوم: زیگزیگ zigzig

  این حالتی است که x با پدر و پدربزرگش در یک راستا قرار گرفته است. در این حالت x با دو دوران به بالا می‌رود.

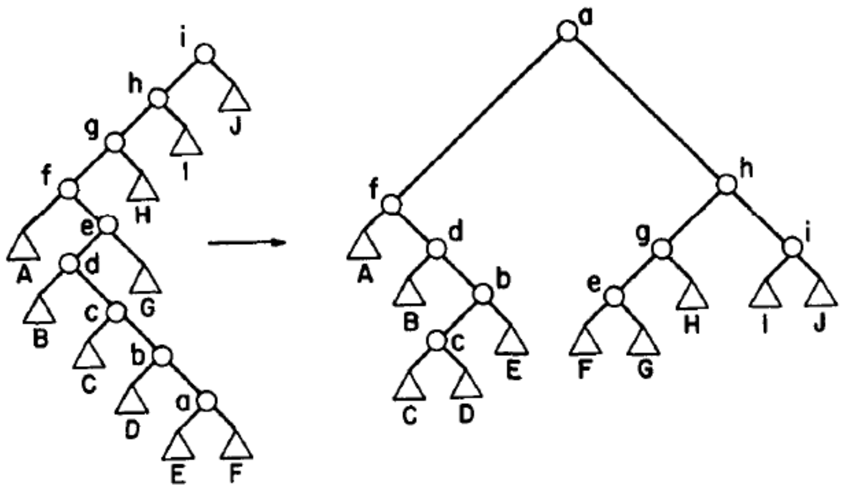
(b)

• حالت سوم: زیگزاگ zigzag

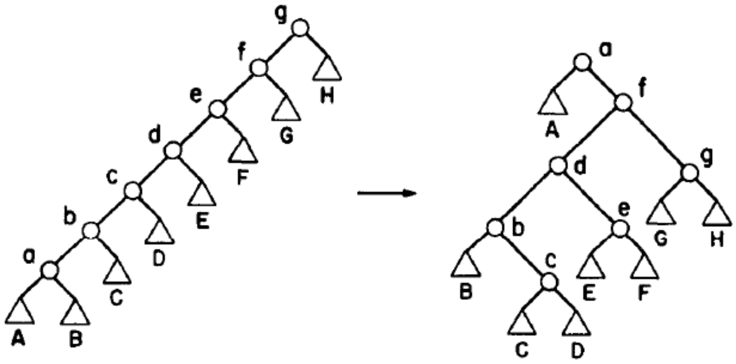
این حالتی است که x با پدر و پدربزرگش بصورت زیگزاگ قرار گرفته است. در این حالت هم x با دو دوران به بالا می‌رود.

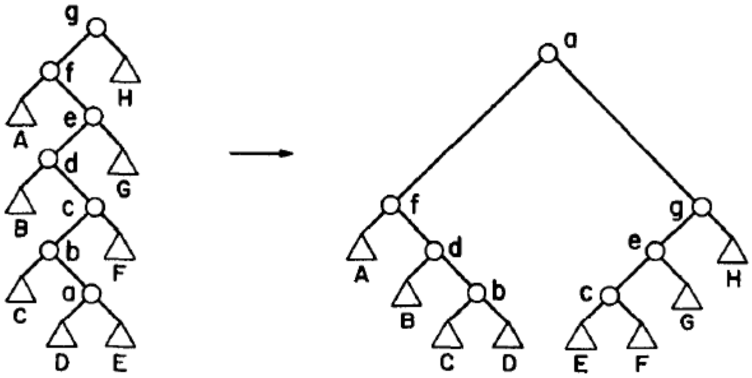
(c)

در عمل splay(x) با توجه به موقعیت اولیه x دنباله‌ای از zig و zigzig و zigzag انجام می‌شود تا اینکه x به ریشه برود. البته حالت zig حداکثر یک بار اتفاق می‌افتد. شکلهای زیر نمونه هایی از اسپلی کردن را نشان می‌دهد. توجه کنید فقط وضعیت اولیه و نهایی درخت رسم شده. در هر سه مورد راس a اسپلی شده است.

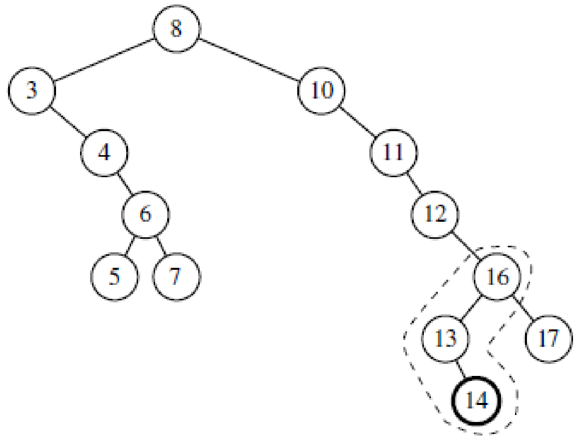


Splaying at node a.

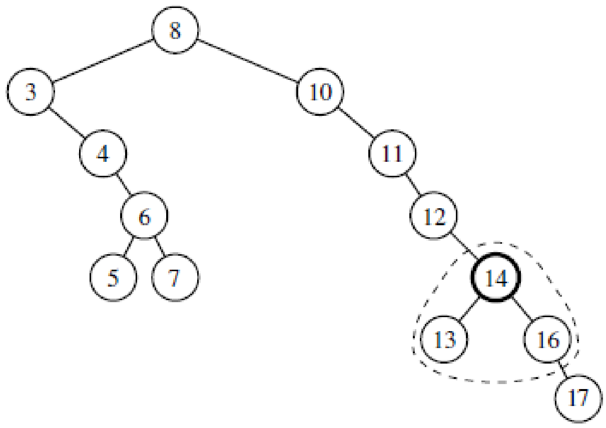




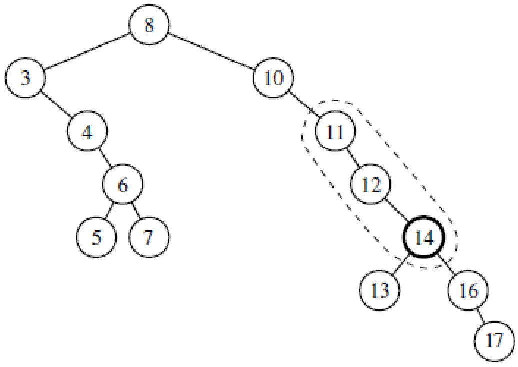
یک مثال کامل از اسپلی کردن در شکل های زیر آمده است.



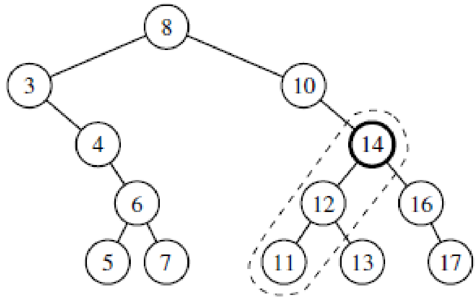
(a)



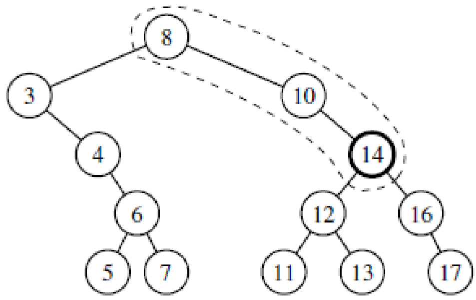
(b)


8
3
10
4
11
6
12
5
7
14
13
16
17

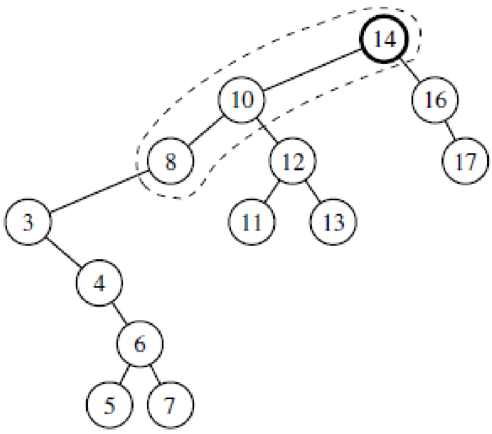
(c)


8
3
10
4
14
6
12
16
5
7
11
13
17

(d)


8
3
10
4
14
6
12
16
5
7
11
13
17

(e)


14
10
16
8
12
17
3
11
13
4
6
5
7

(f)

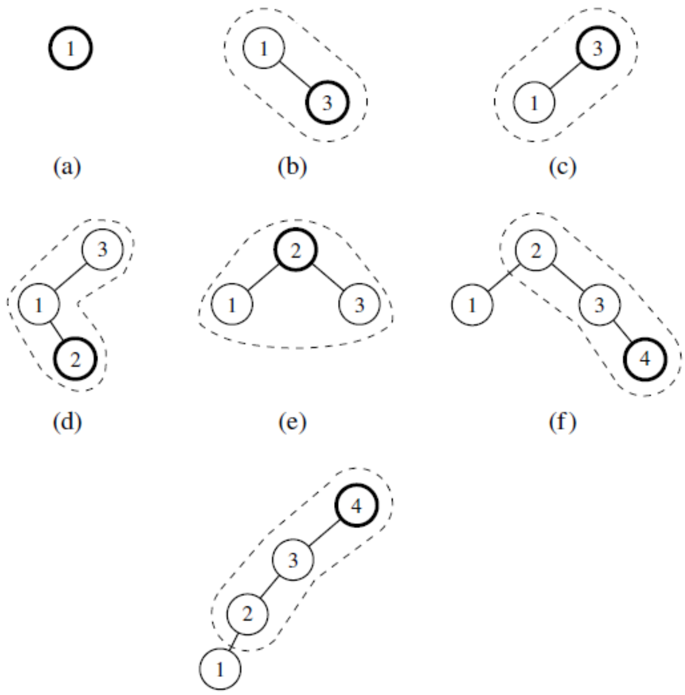
۲.۱ اعمال اصلی Splay Tree

- Search(x)

مانند BST از ریشه شروع به حرکت می‌کنیم تا به x برسیم. سپس x را اسپلی می‌کنیم. اگر x یافت نشد، برگی که در آن عمل جستجو خاتمه یافته است را اسپلی می‌کنیم.

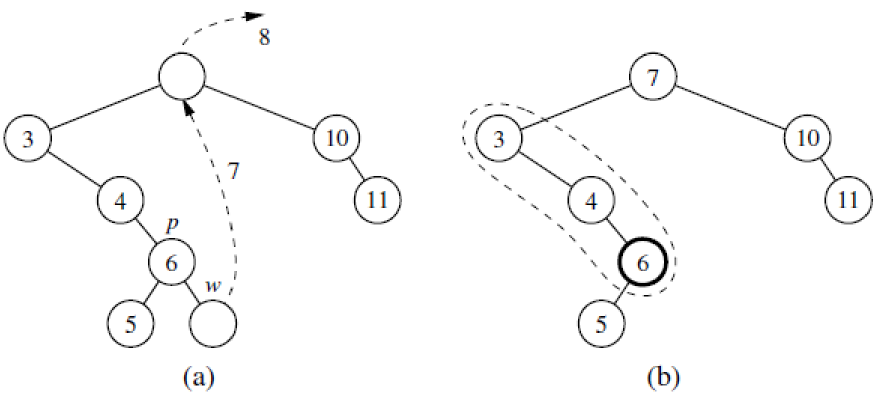
- Insert(x)

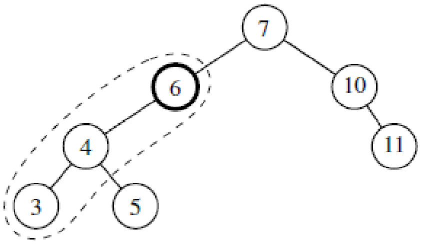
مانند درج در BST معمولی عمل می‌شود با این تفاوت که بعد از درج، عمل اسپلی روی x انجام می‌شود.



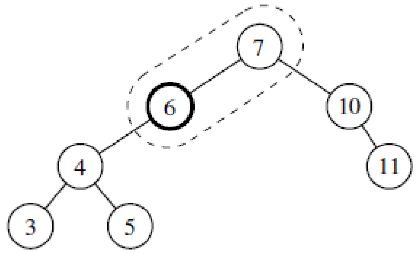
- Delete(x)

در اینجا فرض بر این است که اشاره‌گر به x عنصری که باید حذف شود داریم. اینجا هم مانند حذف در BST معمولی عمل می‌شود. در نهایت پدر راسی که حذف می‌شود اسپلی می‌شود. در شکل زیر عنصر واقع در ریشه درخت حذف شده است. بنا بر قانون حذف در BST عنصر ماکزیمم در زیردرخت سمت چپ جایگزین ریشه می‌شود. لذا در مثال زیر برگ w حذف می‌شود. پس پدر آن p اسپلی می‌شود.

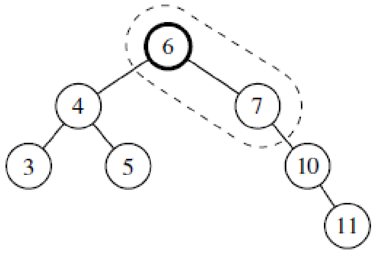




(c)



(d)



(e)

۲ تحلیل سرشکنی درخت اسپلی

مشاهده ۱: فرض کنید $d_T(x)$ عمق راس x در درخت T باشد. زمان splay(x) برابر با $\Theta(d_T(x))$ است.

اثبات: هر قدم زیگ، زیگزیگ یا زیگزاگ به اندازه حداقل یک واحد و حداکثر دو واحد از عمق راس x می‌کاهد. پس تعداد این قدم ها موقع عمل splay(x) حداکثر $d_T(x)$ و حداقل $\frac{1}{2}d_T(x)$ است. علاوه بر این، هر قدم زیگ، زیگزیگ و زیگزاگ زمان اجرایش $O(1)$ است. □

مشاهده ۲: پیچیدگی زمانی اعمال Search(x) و Insert(x) و Delete(x) برابر با پیچیدگی زمانی اسپلی کردنی است که در انتهای آن اعمال انجام می‌شود.

اثبات: می‌دانیم که برای جستجو برای عنصر x باید به اندازه عمق x یعنی $d_T(x)$ پایین برویم (اگر x نباشد به اندازه عمق آخرین برگی که ملاقات می‌شود). در مورد عمل Insert و Delete هم مشابه این است. به اندازه عمق راسی که اضافه یا حذف می‌شود پایین می‌رویم. با توجه به مشاهده قبلی، لذا پیچیدگی زمان اجرای این اعمال متناسب با پیچیدگی زمانی اسپلی کردنی است که اتفاق می‌افتد. □

حال لم زیر را در مورد درخت اسپلی اثبات می‌کنیم. برای اثبات از تکنیک تابع پتانسیل و تحلیل سرشکنی استفاده می‌کنیم.

لم ۳: زمان انجام m جستجوی متوالی در درخت اسپلی حداکثر $O(m\log n + n\log n + m)$ است.

اثبات: فرض کنید عناصر مورد جستجو دنباله Q را تشکیل دهد.

$$Q : \text{Search}(x_1), \text{Search}(x_2), \cdots, \text{Search}(x_m)$$

برای سادگی فرض کنید همه این عناصر در درخت موجود هستند (جستجوی ناموفق نداریم.) دقت کنید لم در حالت کلی صادق است. فقط برای ساده‌تر شدن این فرض را گرفته‌ایم. با توجه به مشاهد ۲ داریم

$$Time(Q) = \Theta(\sum_{i=1}^{m} Time(\text{splay}(x_i)))$$

با فرض اینکه زمان اجرای هر دوران rotation برابر با 1 است، نشان می‌دهیم زمان سرشکنی splay(x_i) حداکثر $3\log n + 1$ است. تاکید می‌کنیم که این تحلیل برای زمان بدترین حالت نیست. زمان واقعی i امین جستجو ممکن است خیلی از اینها بیشتر باشد چون درخت اسپلی لزوما متوازن نیست و همانطور که می‌دانیم زمان جستجو متناسب با عمق عنصر مورد جستجو است.

به عمل splay(x) توجه کنید. می‌دانیم که این عمل ترکیبی از قدمهای زیگ، زیگزیگ و زیگزاگ است. زمان سرشکنی هر قدم را تحلیل می‌کنیم. برای این کار از یک تابع پتانسیل استفاده می‌کنیم. برای معرفی این تابع به یک سری تعریف نیاز داریم.

- به هر عنصر یک وزن غیر صفر و مثبت نسبت می‌دهیم که تغییر نمی‌کند. فرض کنید $w(x)$ وزن عنصر x باشد.
- $s(x)$ را برابر با مجموع وزن عناصری می‌گیریم که در زیردرخت با ریشه x حضور دارند. شامل خود x هم می‌شود.
- رتبه عنصر x یعنی $r(x)$ را برابر با $\log s(x)$ تعریف می‌کنیم.

حال برای درخت T تعریف می‌کنیم

$$\Phi(T) = \sum_{x \in T} r(x)$$

پس پتانسیل درخت T برابر با مجموع رتبه عناصر موجود در T است. حال با توجه به این تعریف، زمان سرشکنی هر قدم را محاسبه می‌کنیم. از فرمول آشنای زیر استفاده می‌کنیم.

$$a_j = c_j + \Phi(D_j) - \Phi(D_{j-1})$$

اینجا c_j زمان واقعی قدم j ام است که می‌تواند یک زیگ، زیگزاگ و یا زیگزیگ باشد. فرض کنید $r(x)$ و $s(x)$ آمار مربوط به حالت قبل از انجام قدم و $r'(x)$ و $s'(x)$ آمار مربوط به بعد از انجام قدم باشد. گزاره زیر را اثبات می‌کنیم.

گزاره ۴: اگر jامین قدم یک zig باشد، داریم $a_j \leq 3(r'(x) - r(x)) + 1$
اگر j امین قدم zigzig یا zigzag باشد داریم $a_j \leq 3(r'(x) - r(x))$

اثبات: هر سه حالت را تحلیل می‌کنیم.

۱. zig

(a)

در حالت زیگ فقط یک دوران رخ می‌دهد. پس $c_j = 1$. باید ببینیم پتانسیل چه تغییری می‌کند. خوشبختانه فقط رتبه عناصر x و y تغییر می‌کند. لذا داریم:

$$a_j = 1 + r'(x) + r'(y) - r(x) - r(y)$$

چون $r(y) \geq r'(y)$ لذا

$$a_j \leq 1 + r'(x) - r(x)$$

چون $r'(x) \geq r(x)$ پس همانطور که ادعا کردیم

$$a_j \leq 1 + 3(r'(x) - r(x))$$

۲. zigzig

(b)

چون دو دوران داریم پس $c_j = 2$. چون فقط رتبه عناصر x و y و z تغییر می‌کند لذا داریم

$$a_j = 2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z)$$

چون $r'(x) = r(z)$

$$a_j = 2 + r'(y) + r'(z) - r(x) - r(y)$$

چون $r'(x) \geq r'(y)$ پس

$$a_j \leq 2 + r'(x) + r'(z) - r(x) - r(y)$$

چون $r(x) \geq r(y)$ پس

$$a_j \leq 2 + r'(x) + r'(z) - 2r(x)$$

می‌خواهیم نشان دهیم $a_j \leq 3(r'(x) - r(x))$ برای این منظور کافی است نشان دهیم

$$2 + r'(x) + r'(z) - 2r(x) \leq 3(r'(x) - r(x))$$

به عبارت دیگر

$$2r'(x) - r(x) - r'(z) \geq 2$$

که معادل است با

$$r(x) + r'(z) - 2r'(x) \leq -2$$

با توجه به تعریف $r(x) = \log s(x)$ داریم

$$r(x) + r'(z) - 2r'(x) = \log s(x) + \log s'(z) - 2\log s'(x) = \log \frac{s(x)}{s'(x)} + \log \frac{s'(z)}{s'(x)}$$

چون $s'(z) + s(x) \leq s'(x)$ پس $\frac{s(x)}{s'(x)} + \frac{s'(z)}{s'(x)} \leq 1$. در نتیجه

$$\log \frac{s(x)}{s'(x)} + \log \frac{s'(z)}{s'(x)} \leq -2$$

این از اینجا ناشی می‌شود که برای دو عدد مثبت a, b وقتی $a + b \leq 1$ آنگاه $\log a + \log b \leq -2$. این ادعای ما را ثابت می‌کند.

۳. zigzag

(c)

مشابه حالت قبل چون دو دوران داریم پس $c_j = 2$. چون فقط رتبه عناصر x و y و z تغییر می‌کند لذا داریم

$$a_j = 2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z)$$

چون $r'(x) = r(z)$ و $r(x) \leq r(y)$ داریم

$$a_j \leq 2 + r'(y) + r'(z) - 2r(x)$$

ادعا می‌کنیم $a_j \leq 2(r'(x) - r(x))$. برای این منظور کافی است نشان دهیم

$$2 + r'(y) + r'(z) - 2r(x) \leq 2(r'(x) - r(x))$$

به عبارت دیگر

$$r'(y) + r'(z) - 2r'(x) \leq -2$$

مانند حالت قبل

$$\log s'(y) + \log s'(z) - 2\log s'(x) \leq -2$$

که معادل است با

$$\log \frac{s'(y)}{s'(x)} + \log \frac{s'(z)}{s'(x)} \leq -2$$

چون $s'(y) + s'(z) \leq s'(x)$ مانند حالت قبل نتیجه می‌گیریم که این نامساوی درست است و لذا ادعای ما ثابت می‌شود.
این اثبات گزاره را کامل می‌کند. □

حال به splay(x) برمی‌گردیم. فرض کنید splay(x) دنباله‌ای از قدمهای زیر باشد.

$$z_1, \cdots, z_k$$

هر z_j یک عمل زیگ، زیگزیگ یا زیگزاگ است. فرض کنید درخت قبل انجام اسپلی T و r تابع رتبه آن باشد. درخت بعد از قدم j را T_j می‌نامیم و r_j را تابع رتبه بعد از قدم j ام در نظر می‌گیریم. اگر $a(\text{splay}(x))$ زمان سرشکنی splay(x) باشد داریم

$$a(\text{splay}(x)) = a(z_1) + a(z_2) + a(z_3) + \cdots + a(z_k)$$

با توجه به گزاره ۴ و این مشاهده که فقط آخرین قدم می‌تواند یک زیگ باشد، داریم

$$a(\text{splay}(x)) \leq 3(r_1(x) - r(x)) + 3(r_2(x) - r_1(x)) + 3(r_3(x) - r_2(x)) \cdots + 3(r_k(x) - r_{k-1}(x)) + 1$$

بعد حذف تسکلوپی عبارات بدست می آید

$$a(\text{splay}(x)) \leq 3(r_k(x) - r(x)) + 1$$

اگر t ریشه درخت T باشد، چون x در نهایت در ریشه قرار گرفته است داریم $r_k(x) = r(t)$ در نتیجه

$$a(\text{splay}(x)) \leq 3(r(t) - r(x)) + 1 = 3 \log \frac{s(t)}{s(x)} + 1$$

حال برای اینکه ثابت کنیم $a(\text{splay}(x)) = 3 \log n + 1$ کافی است برای هر عنصر x قرار دهیم

$$w(x) = \frac{1}{n}$$

اگر t ریشه باشد، بدست می‌آید $s(t) = 1$ و در بدترین حالت $s(x) = \frac{1}{n}$. در نتیجه

$$a(\text{splay}(x)) \leq 3 \log \frac{s(t)}{s(x)} + 1 = 3 \log \frac{1}{1/n} + 1 = 3 \log n + 1$$

در نهایت برای دنباله اعمال $Q' : \text{splay}(x_1), \text{splay}(x_2), \cdots, \text{splay}(x_m)$ و با توجه به رابطه

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} a_i + \Phi(D_0) - \Phi(D_m)$$

داریم

$$\sum_{i=1}^{m} Time(\text{splay}(x_i)) = \sum_{i=1}^{m} a(\text{splay}(x_i)) + \Phi(D_0) - \Phi(D_m)$$

از بحثهای بالا نتیجه می‌شود

$$\sum_{i=1}^{m} Time(\text{splay}(x_i)) \leq m(3 \log n + 1) + \Phi(D_0) - \Phi(D_m)$$

پتانسیل چقدر می‌تواند کاهش یابد؟ به عبارت دیگر $\Phi(D_0) - \Phi(D_m)$ حداکثر چقدر است؟ اگر $y_1, \ldots, y_n$ عناصر موجود در درخت باشند و تعریف کنیم $W = \sum_{i=1}^{n} w(y_i)$ میزان کاهش پتانسیل حداکثر می‌تواند

$$\sum_{i=1} r_0(y_i) - r_m(y_i) = \sum_{i=1}^{n} \log \frac{s_0(y_i)}{s_m(y_i)} \leq \sum_{i=1}^{n} \log \frac{W}{\min\{w(y_i)\}} = \sum_{i=1}^{n} \log \frac{1}{1/n} = n \log n$$

نهایتا بدست می‌آید

$$\sum_{i=1}^{m} Time(\text{splay}(x_i)) \leq m(3 \log n + 1) + n \log n$$

$\square$