بخش اول : تحلیل پیچیدگی زمانی

الف) پیچیدگی زمانی

ساختار کلی سیستم

سیستم از چهار بخش اصلی تشکیل شده است:

۱ .پردازش **فرمول‌ها** (ارزیابی، infix به postfix ، Tokenization)

۲ .**مدیریت وابستگی‌ها** (گراف وابستگی، تشخیص دور)

۳ .**محاسبه مجدد** (به‌روزرسانی زنجیره‌ای)

۴ .**مدیریت تاریخچه** (Undo/Redo)

1. ارزیابی یک فرمول (Evaluate formula): الگوریتم خطی است و هر کاراکتر فرمول دقیقا یک بار پردازش میشود.
   - تبدیل infix به postfix : O(L) که L طول فرمول است.
   - محاسبه postfix : O(L) برای پردازش توکن ها.
   - جمع کل: O(L): **توضیح :**الگوهای regex به صورت خطی روی فرمول اجرا می‌شوند.

2. استخراج وابستگی ها (Extract Dependencies)
   - جستجوی الگو در فرمول با استفاده از regex : O(L)

3. محاسبه مجدد وابستگی ها (Recalculate Dependencies)

- یافتن سلول های وابسته : (BFS) که: O(V+E)
  - تعداد سلول ها <- V= R*C
  - **تعداد یال های گراف وابستگی E**
- مرتب سازی توپولوژیکی : O(V+E)
- محاسبه هر سلول وابسته : O(L)
- پیچیدگی زمانی : O(V+E+L)

4. بدترین حالت

اگر یک سلول به تمام سلول های دیگر وابسته باشد (مثل =SUM(A1:D1))

- V=R*C
- E=V-1=V (هر سلول به همه وابسته است)
- پیچیدگی زمانی: O(V*L)=O((R*C)*L)

5. دستور N برای SET

- هر دستور SET ممکن است کل گرید را به روز کند.
- بدترین حالت ممکن: O(N*(R*C)*L)

6. مدیریت گراف وابستگی

- یافتن سلول های وابسته (BFS)
- V تعداد رئوس (سلول ها) : R*C
- E : تعداد یال ها (وابستگی ها)
- پیچیدگی زمانی: O(V+E)

ب) پیچیدگی فضایی

1. ذخیره سازی گرید

- آرایه دو بعدی سلول ها: O(R*C)
  - حافظه هر Cell :
    - ✓ rawContent: String ≈ L . بایت
    - ✓ computedValue: Object ≈ ۸ بایت(reference)
    - ✓ dependencies: Set<String> ≈ ۱۶ بایت + هر رفرنس
    - ✓ سایر فیلدها: ≈ ۲۰ بایت

کل گرید: O(R*C*L)

2. گراف وابستگی

- لیست مجاورت: O(V+E)
- در بدترین حالت(هر سلول به همه وابسته است) : O(R*C)

3. پشته و صف

- پشته برای ارزیابی فرمول : O(L)
- صف برای BFS: O(V)

4. تاریخچه (UNDO/REDO)

- ذخیره H حالت: O(H*R*C)

5. جمع کل فضایی

بدترین حالت: O((R*C) + (H*R*C) + L)

بخش دوم: روش های بهینه سازی

الف) جلوگیری از محاسبه مجدد کل گرید

- روش فعلی: فقط سلول های وابسته محاسبه میشوند.
- الگوریتم:
  - ♦ هنگام تغییر یک سلول، با BFSتمام سلول هایی که به آن وابسته اند پیدا می شوند.
  - ♦ فقط آن سلول ها محاسبه مجدد می شوند.
- اثر :کاهش از O(R × C) به O(K)که Kتعداد سلول‌های وابسته است

ب) محاسبه مجدد تنها سلول های وابسته پس از تغییر یک سلول

مکانیزم: ۱ .هنگام تعریف فرمول، وابستگی‌ها استخراج و ذخیره می‌شوند.

۲ .برای هر سلول، لیست dependentsنگهداری می‌شود.

۳ .هنگام تغییر، فقط این لیست دنبال می‌شود.

- گراف وابستگی:
  - هر سلول لیستی از سلول هایی که به آن وابسته اند (Dependents) دارد.
  - هنگام تغییر، فقط این مسیرها دنبال شوند.
- مثال :اگر C1 = A1 + B1باشد، فقط C1پس از تغییر A1یا B1محاسبه می‌شود.

ج) استفاده از ساختارهای داده های مناسب

1. صف برای مدیریت محاسبات

- مشکل: اگر سلول ها به ترتیب اشتباه محاسبه شوند، مقادیر قدیمی استفاده می شوند.
- راه حل: مرتب سازی توپولوژیکی با صف
  - سلول هایی که هیچ وابستگی حل نشده ای ندارند، اول محاسبه می شوند.
  - پیچیدگی O(V + E) :به جای جستجوی کامل.

2. بهینه سازی گراف

- ذخیره هم زمان Dependent , Dependencies :

- Dependencies : سلول هایی که این سلول به آن ها نیاز دارد.
- Dependent : سلول هایی که به این سلول نیاز دارند.

- افزودن سریع و حذف سریع

3. استفاده از Hash Maps برای دسترسی سریع

- دسترسی به سلول با آدرس O(1) :با مپ .CellReference → Cell
- جستجوی وابستگی‌ها O(1) :برای بررسی وجود ارجاع.

4. پشته برای ارزیابی فرمول‌ها

الگوریتم:Shunting-yard

- . تبدیل Infix به Postfix با حفظ اولویت‌ها
- پشتیبانی از پرانتز و عملگرهای یونار‌ی
- پیچیدگی خطی O(L) :

د) ذخیره سازی مقادیر حساب شده برای کاهش مصرف حافظه و دسترسی سریع تر.

1. Caching .مقادیر سلول‌ها

- مشکل: فرمول های تکراری چندبار محاسبه می شوند.
- راه‌حل :ذخیره نتیجه فرمول با کلید .(formula, dependencies_values)

2. Memoizationبرای محدوده‌ها

- برای توابع تجمعی مثل SUM(A1:D1)
  - نتیجه را cache کنید.
  - اگر سلولی در محدوده تغییر کرد، فقط تفاوت را اعمال کنید (Incremental Update).
  - مثالnewSum = oldSum - oldValue + newValue :

3. Lazy Evaluation

- . مشکل :محاسبه همه سلول‌ها حتی وقتی نیازی نیست.
- راه‌حل :فقط وقتی سلولی خواسته شد (مثلاً برای نمایش) محاسبه شود.
- پیاده‌سازی :یک فلگ isDirtyبرای سلول‌هایی که نیاز به محاسبه مجدد دارند.

4. Batch Updates
   - همه تغییرات را جمع‌آوری کنید.
   - یک بار محاسبه مجدد انجام دهید.
   - از محاسبات تکراری جلوگیری می‌کند.