Onur Demir 18011073

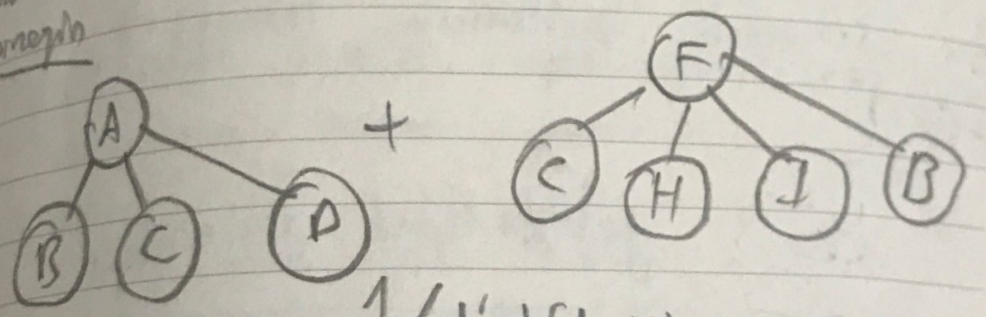
1) Kaç tane graf arıtleri oluştur,
2) Iki graf birleştirecek algoritmayı yap.

3) Arama sorgusunda çevresindeki düğümleri gezdir.

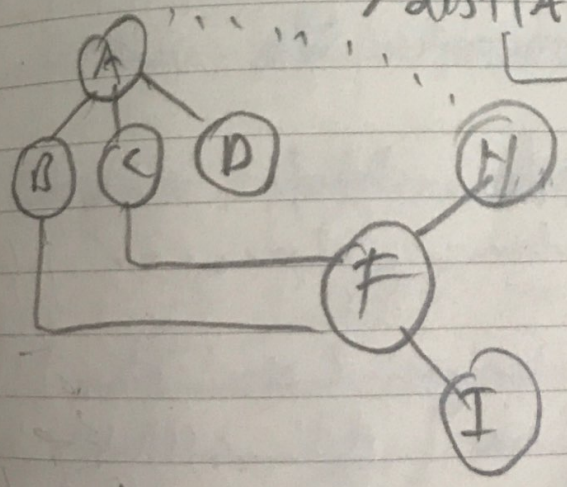
2kg Graf nasıl tutulacak? B. Matris gösterimi
B. Liste gösterimi

1) B. Matrisi kullanırsak kullanıcı kendi ormanıyla ilgili olmayanları da sorgulayabilir. (Bellek ilgili olmama durumuna göre bir olasılık katsayı atanabilir.) mask matrisi olarak 0 olan yerler var.

Örneğin



1/dist(A,H) → ?



Bitişiklik matrisi kullanırsak proje gelişmeye açık olabilir. Neden?

A ve H arası normal de matris gösterimde 0 dır ama olasılıksal maskelenmiş matris gösteriminde A→H satır sütununa gelen rasma 1/dist(A,H) gibi bir katsayı atanabilir.

2) Bitişiklik listesi gösterimi graf gösterimanda kullanılabilen

Bu işlem yükünü hafifletecektir. (sparse matrix ağırlığından kurtaracaktır.

Arama ilişkileri yoğun bir ilişki olmadığında bu şekilde graf oluşturulabilir

3) Ek bir fonksiyon bitişiklikler listesi gösterimadan matrise çevirerek 1 deki gibi matris istenen gerçekleştirmesi sağlanabilir.
(Bu seçenek kullanıcının isteği ile oalıp kalpanabilir.)

1 ve 3 geliştirilme seçenekleri

Sorun: Graf oluşturacak ilk veri nasıl verilecek

- Programın tarafından hazır oluşturulan bitişiklik matrisinin bir anlamı olmaz,

Çözüm: Bir metin belgesi (veri tabanı) ikili kelimeleri tutar. Program da bu ikili çiftlerden bitişiklik matrisi oluşturabilir.

Örn: pandemic ; pandemic definitions
pandemic vs epidemic ; pandemic
pandemic in history ; pandemic
pandemic definitions ; pandemic vs epidemic gibi

2) Bitişiklik listesi gösterimi graf gösteriminde kullanılabilir

Bu işlem yükünü hafifletecektir.
(Sparse matrix ağırlığından kurtaracaktır.

Arama ilişkileri yoğun bir ilişki olmadığında bu şekilde graf oluşturulabilir

3) Ek bir fonksiyon bitişiklikler listesi gösteriminde matrise çevirerek 1 deki gibi matris işlemi gerçekleştirmesi sağlanabilir.
(Bu seçenek kullanıcının isteği ile oluşturulabilir.)

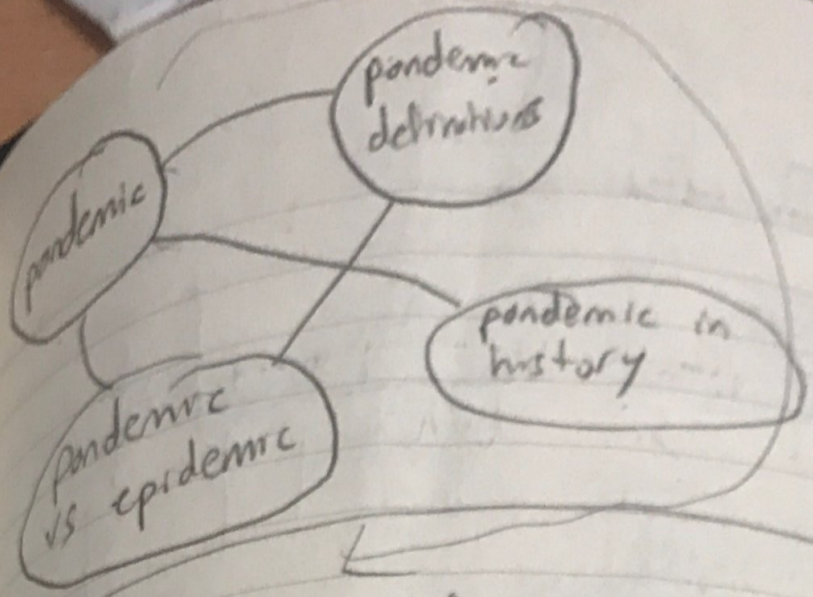
1 ve 3 geliştirilme seçenekleri

Sorun: Graf oluşturacak ilk veri nasıl verilecek

- Programın tarafından hazır oluşturulan bitişiklik matrisinin bir anlamı olmaz.

Çözüm: Bir metin belgesi (veri tabanı) ikili çiftleri tutar. Program da bu ikili çiftlerden bitişiklik matrisi oluşturabilir.

Örn: pandemic ; pandemic definitions
pandemic vs epidemic ; pandemic
pandemic in history ; pandemic
pandemic definitions ; pandemic vs epidemic gibi



pandemic → pandemic def, pandemic in hist, pand. vs. ep
pan. def → pan. vs. ep, pand.
pan. in hist → pand.
pan. vs. epi. → pand., pand. definitions.

program bu şekle dönüştürmeli.

Sorunlar 1) metin içinde yazım sırası ve birden fazla tekrarlanma program hazımamalı ve fazla satırı temizlemeli.

2) Bu input alımı ve tutulma şekli çok karmaşık. (Muhtemelen sistem açılıp kapandığında tekrar hesaplanması gerektiğinden sistemi yavaşlatacak.)

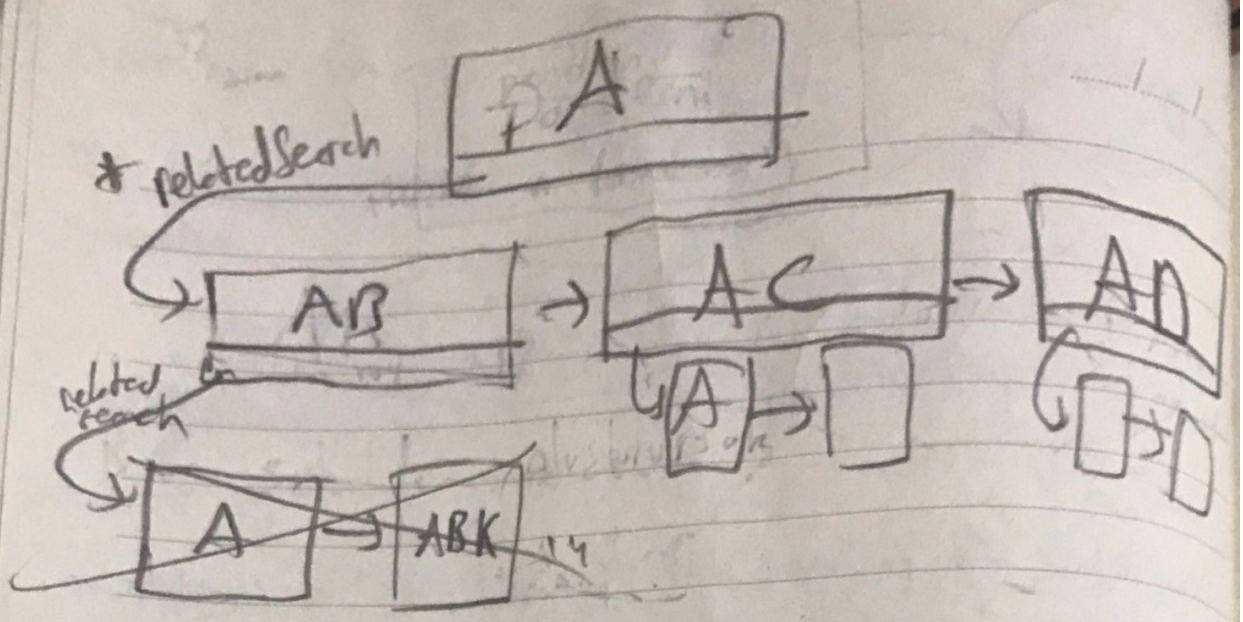
Çözüm: Verinin tutulması ve verinin alınması ayrıştırılmalı.

Yapı nasıl olmalı?

search
related_search*

search ve related_search odada iki değişken bulunmalı - search char* lardan oluşur.

related search



Bize sadece A arandığındaki sonuçlar ama

AB nin aranmesi A nın aranmasından bağımsız olmalı

pandemic in history arandığında ilgili aramalarda pandemic olmayabilir ama plaque olabilir

2 yaklaşım olabilir

1) İlgili aramalar örneğindeki gibi aranan kelime nin devamı olabilir

pandemic → pandemic in history

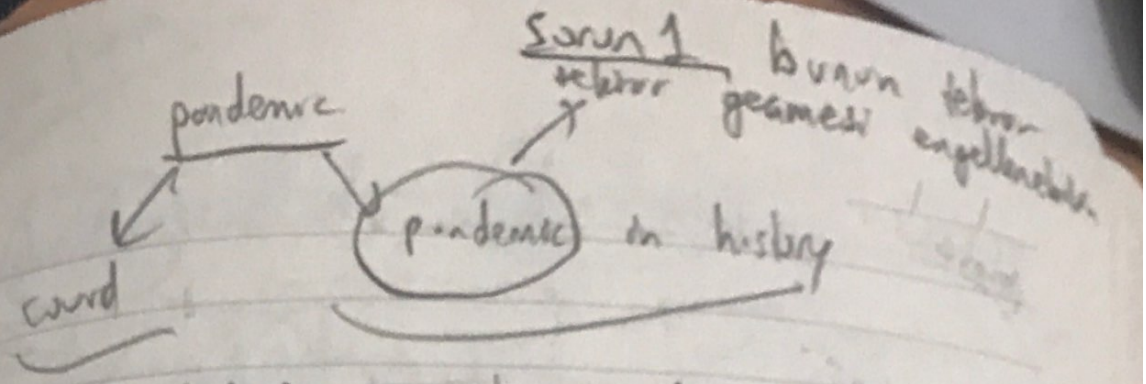
2) " ... gibi

2) Ek bir kelime de olabilir (1'le bahsettiğim olasılıksal bir matriste belli bir eşik değeri üstü yazılabilir.

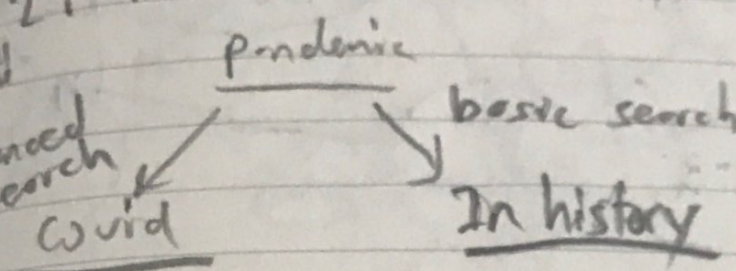
pandemic → card

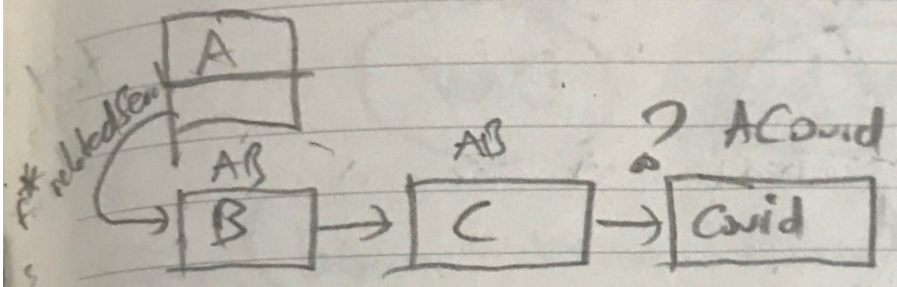
pandemic → pandemic in history

tekrar tekrar geçiyor



Sorun 2: related search ikiye farklı durum bölündü

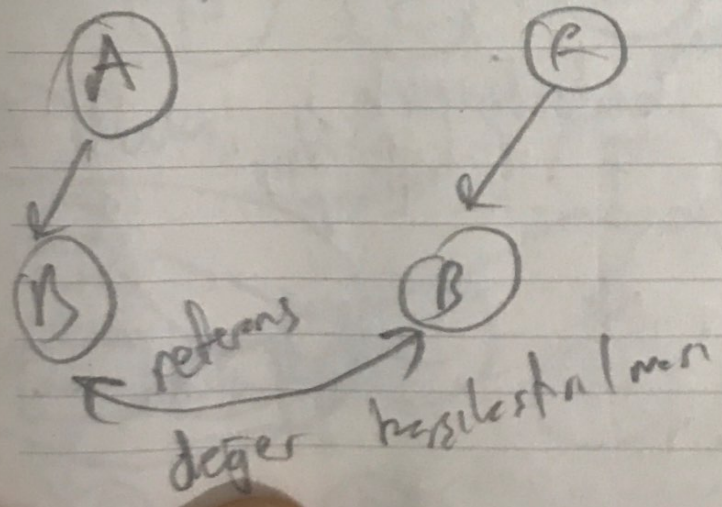




bu struct farklı şekilde tasarlanmalı.

Sorun: Related Search değişkenleri bir düğüm mü olmalı yoksa string mi?

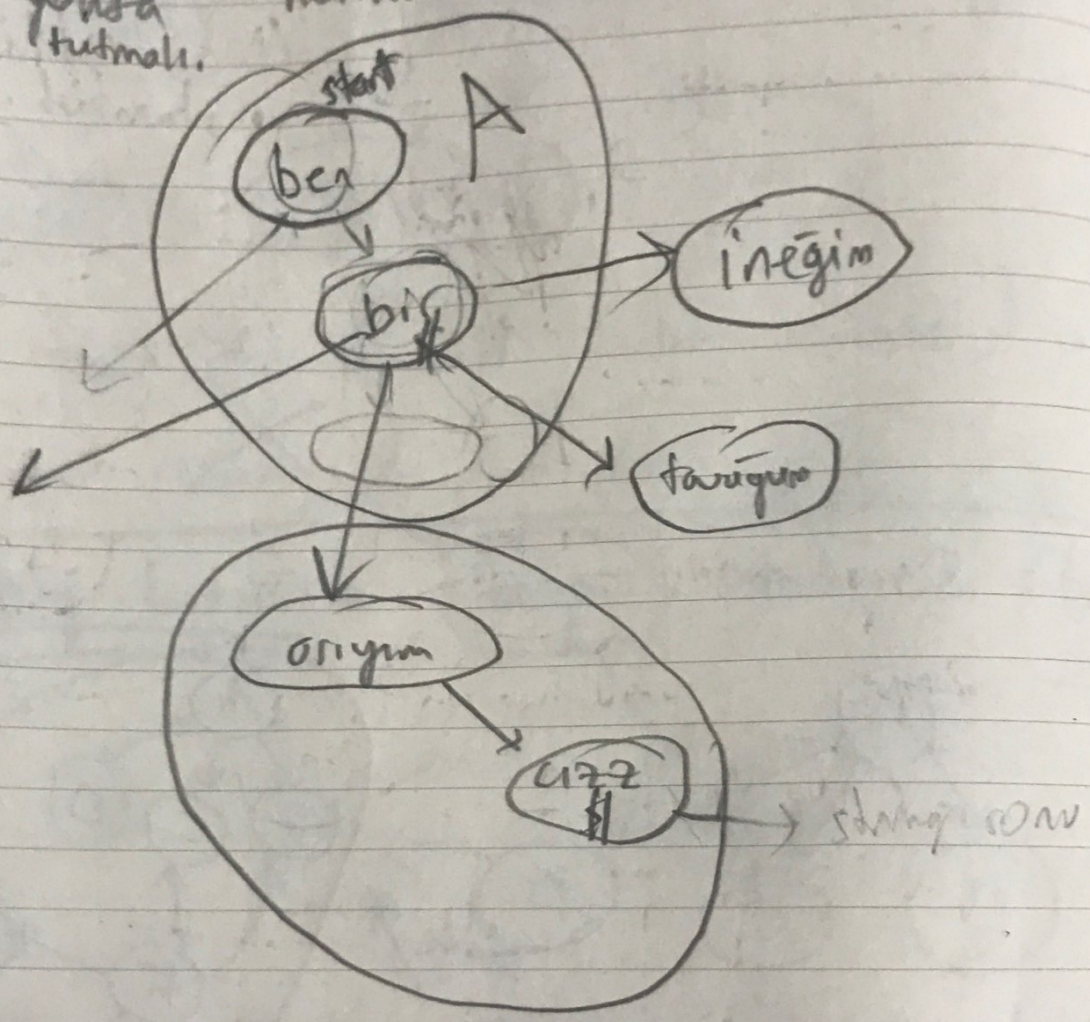
Örneğin pandemic in history düğüm olursa pandemic relatedSearch kısmındaki pandemic in history ile bütün bir node üzerinden aynısı referans kullanılarak karşılaştırılabilir. (Stringler karşılaştırmak uzun süre alacaktır.)



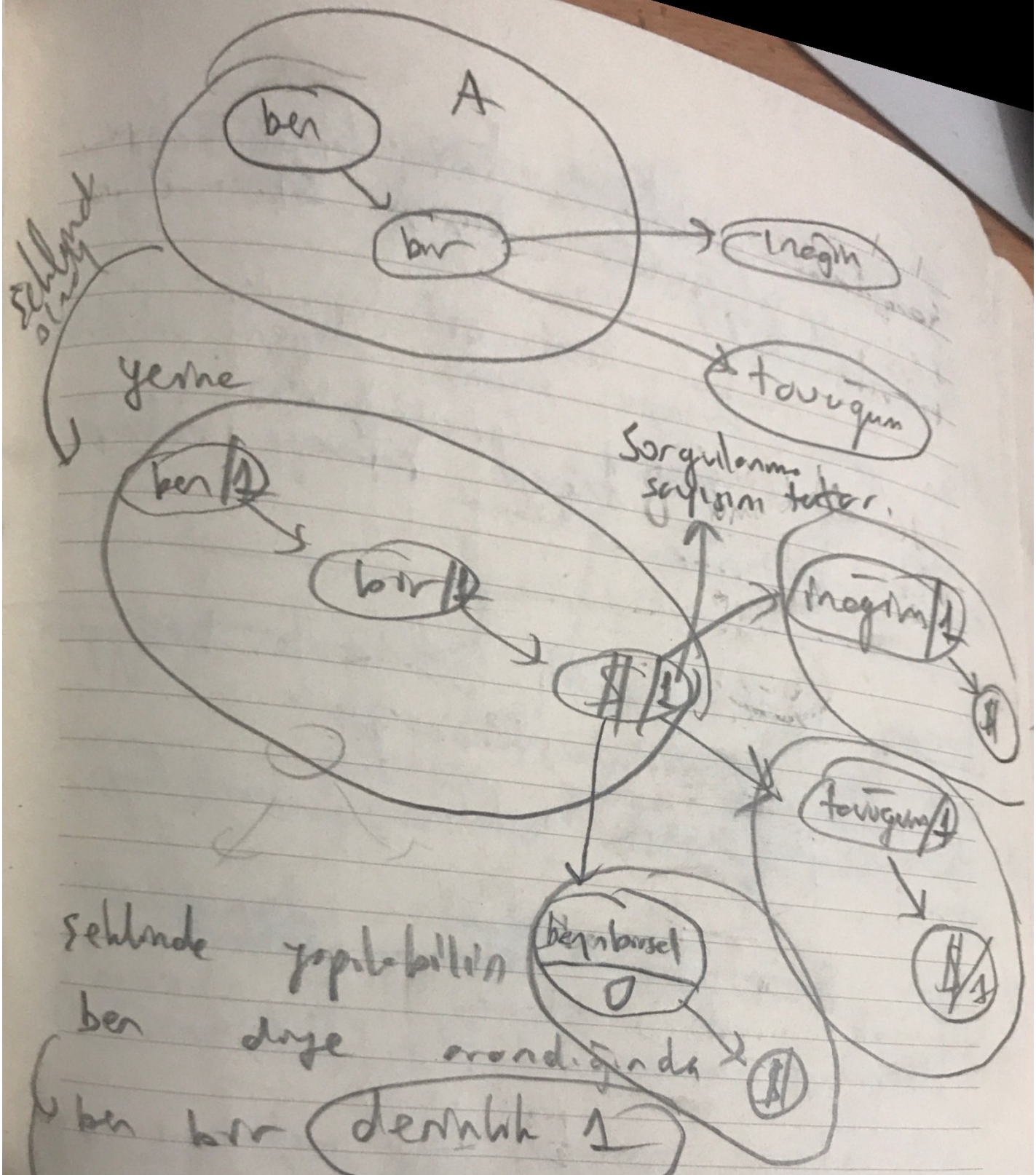
CamScanner ile tarandı

Sorgulama için bağlantılı nodel
bilinmeli bizim durumumuzda A ve
F den
Bunlar bir dizide tutulmalı

Node lar ilgili kelimenin prefix tree de
yoksa normal kelime mi olduğu bil
tutmalı.



* String sonları bir şekilde belirtilmeli
bu şekilde en yakın kelimeler
BFS ile bulurabiliriz



şeklinde yapabilirim

ben daye arandığında

ben bir derinlik 1

ben bir inegim derinlik 2

ben bir inegim derinlik 2

ben birsel derinlik 2

geveplerim bits sayesinde nem

Sorunlar: Kısa ben diye giriş yaptığında hangi sayma değeri artırılacak?

1) en yakın stringin sayma değeri artırılabilir

2) yeni bir # değeri kelimeye eklenip değeri artırılır

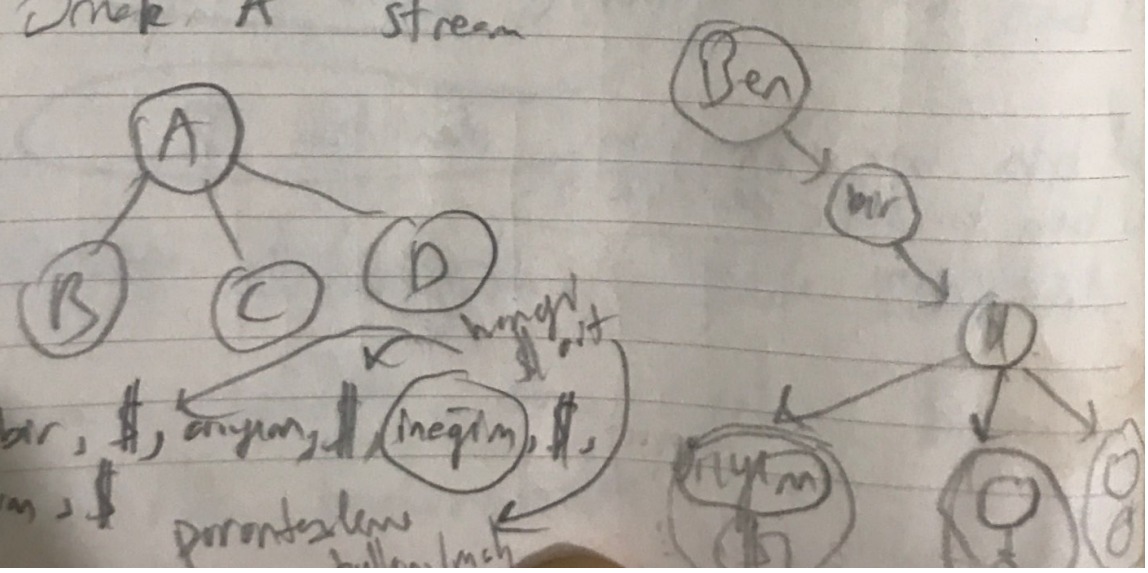
Sorun 1) Veri nasıl tutulacak başlangıçta nasıl bir veri duşturacağız.

* Arama bilgisinin tutulacağı bir log olmalı.

* Eğer bunu bir stream şeklinde yaparsak A stream F stream oluşturup, bunda

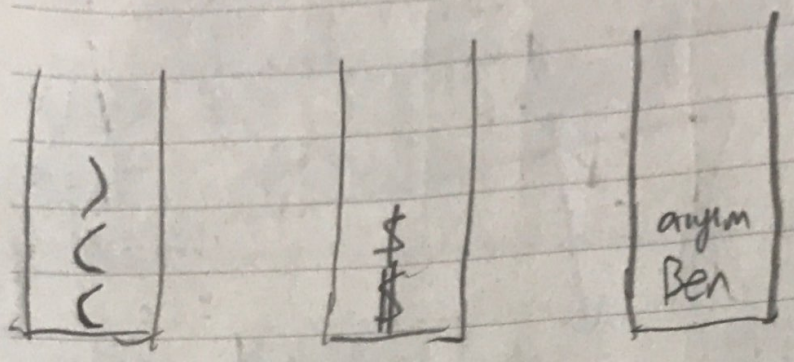
AF stream merge ile metne yazılabilir

Örnek A stream



(Ben, bir, $, (arıyım, $), (inegim, $

bu stream okumak için push-d
otomat kullanılabilir.



1. (
2. Ben
3. $
4. (
5. arıyım
6. $
7.)

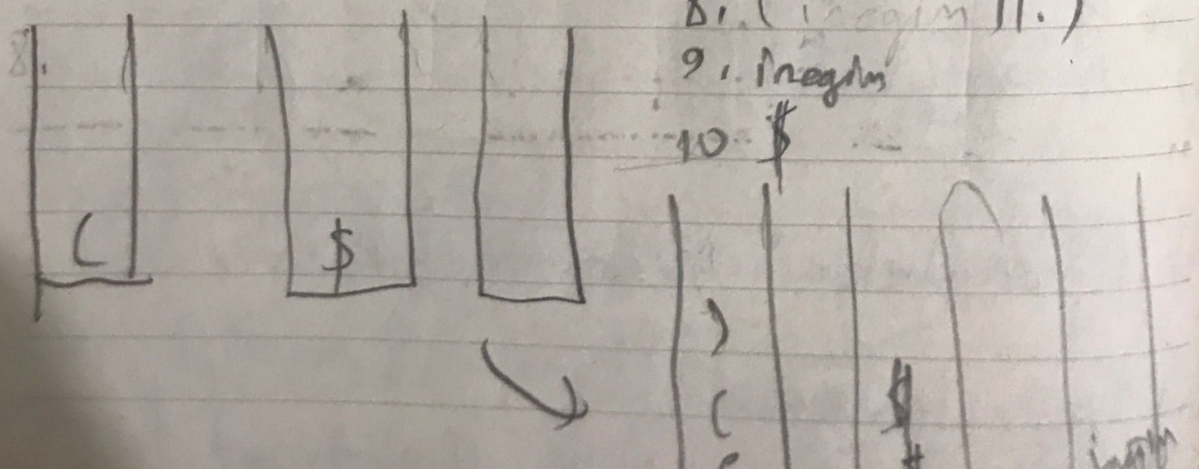
bu şekilde eklendikten sonra ilk) gelince ilk stoktan 2 pop

ikinci stoktan 1 pop 1 peek

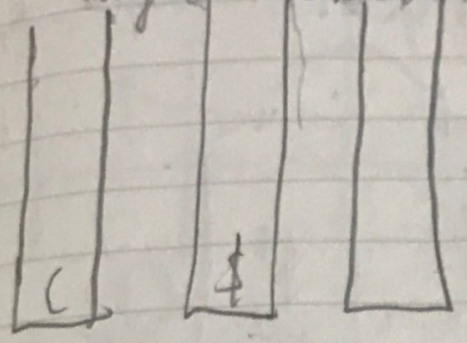
"üçüncü" stoktan boşalana kadar pop

peek yapılan değerin neighbourunu üçüncü
stoktaki değerler yazılır

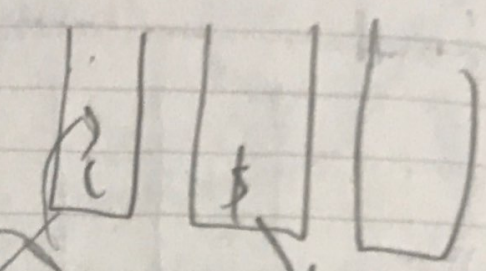
8. ((inegim 11.)
9. inegim
10. $



Inegim eklenir



11.) eklenir

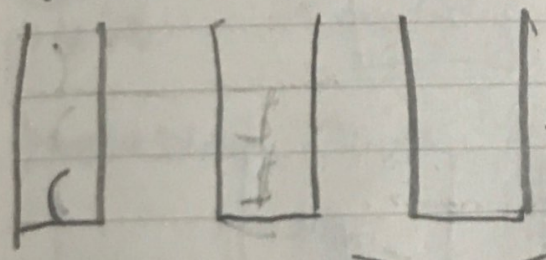


çıkarıldı
pop

peek edildiğinde eleman yok bitti

Sorun
Stream bu şekilde okunabilir ama nasıl tree haline getirilebilir

Yeni yaklaşım
yine 3 stack olacak.

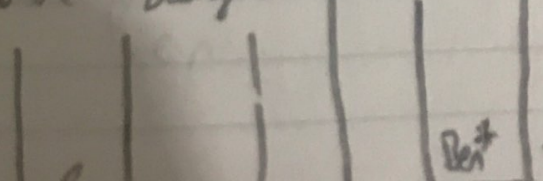


Bu iki stack pointer adreslerini tutacak.

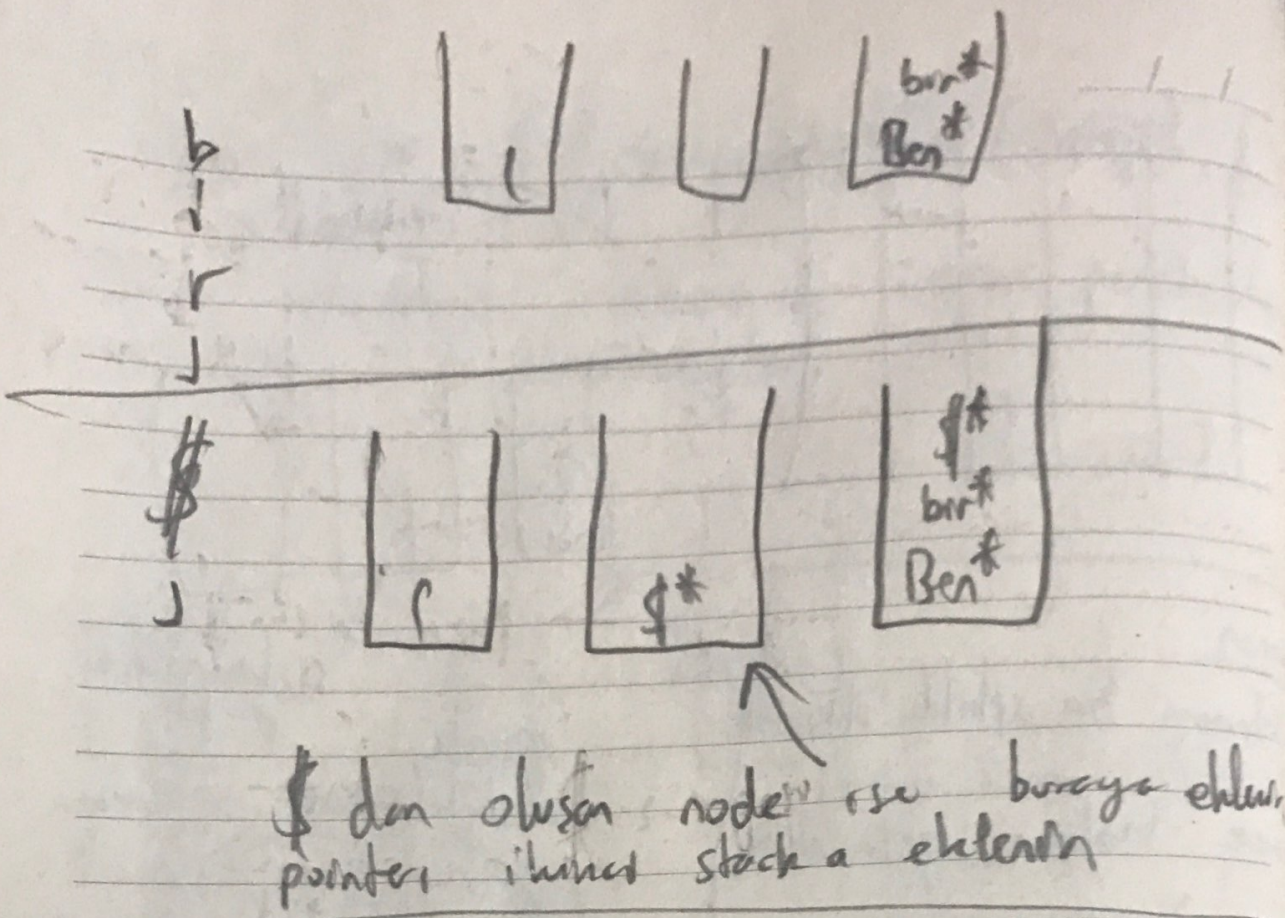
1. C
2. Ben
3. e
4. n
5. $ → geldiğinde

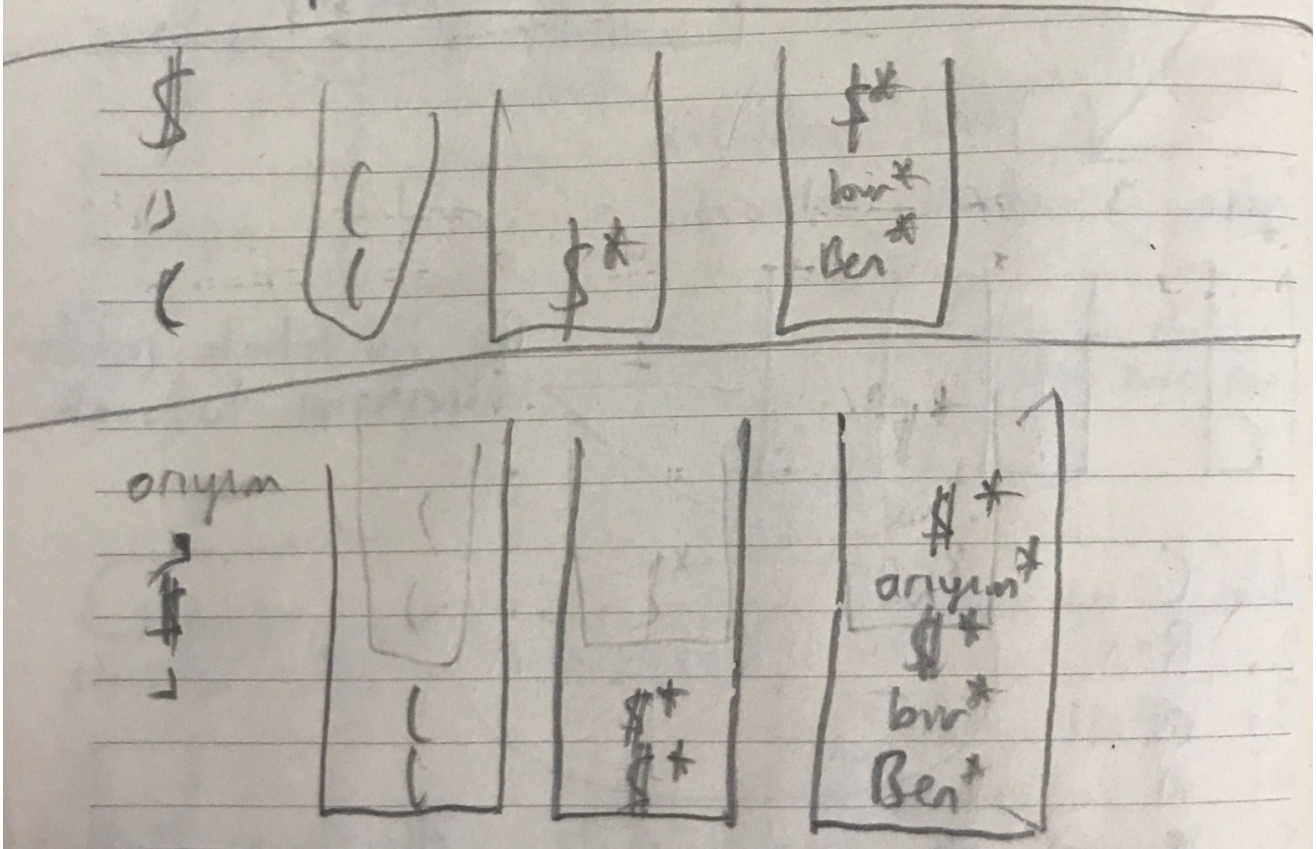
link B → e → n
1.57
count

Bir tane düğüm oluştur Ben'i düğüme ata düğümüne Ben düğümünü 3. stack a koy



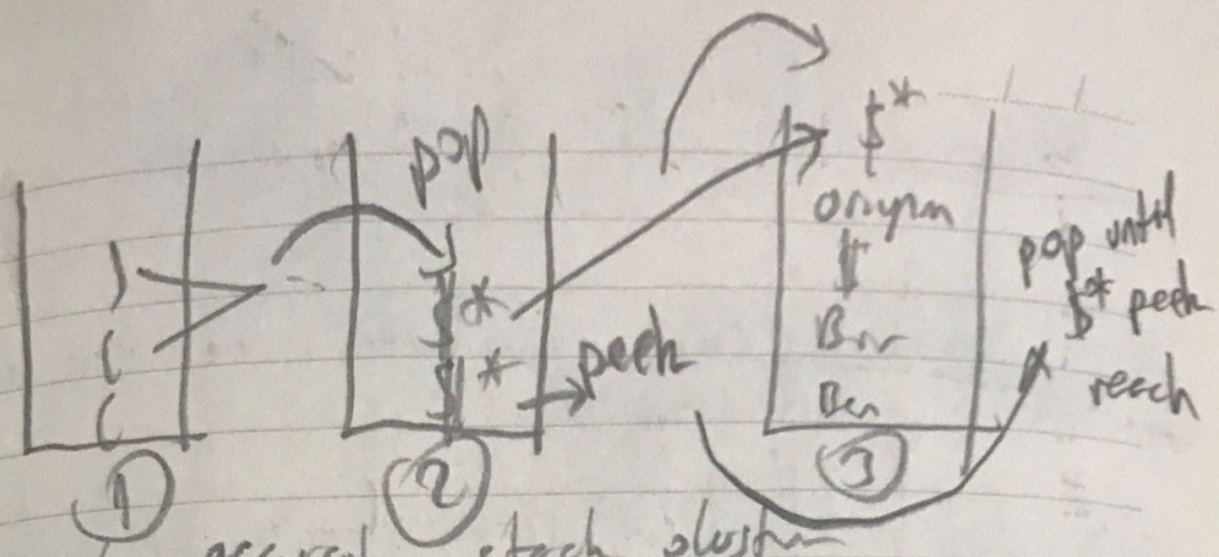
→ TreeNode SZ



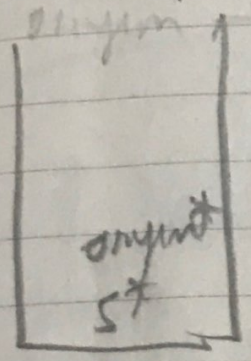


60960

CamScanner ile tarandı



yeni geçici stack oluştur

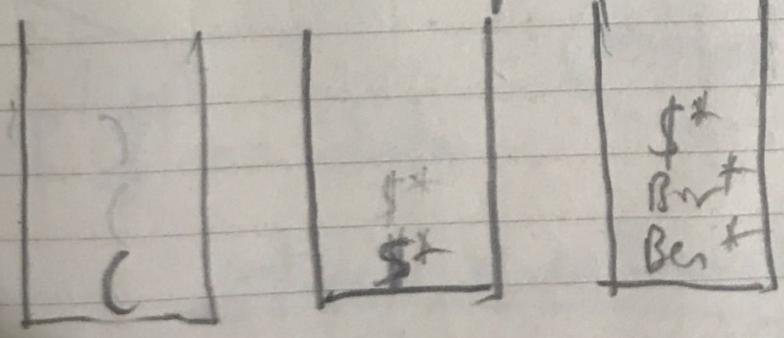


start* onyum → $* → itr$

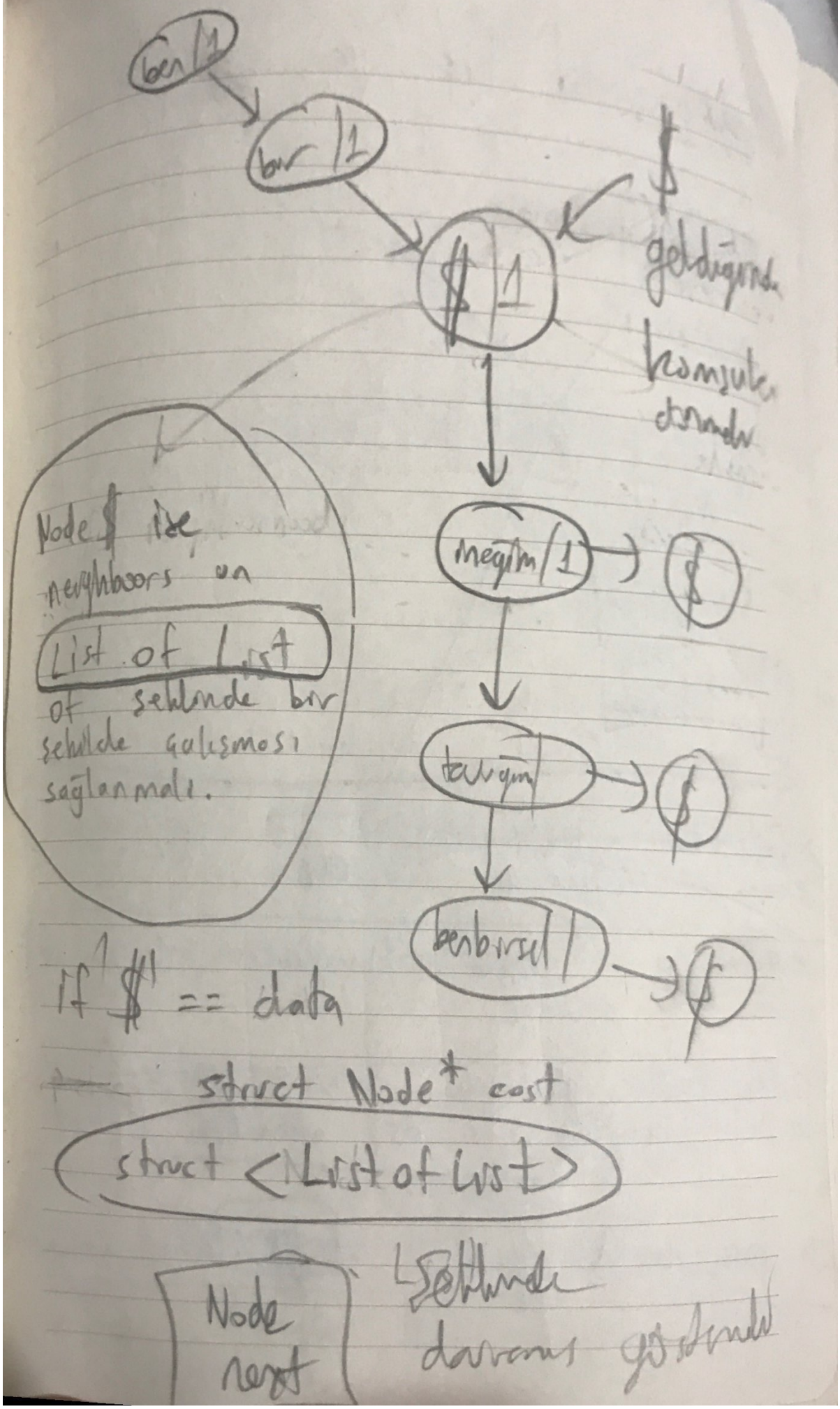
itr itr

onyum void pointer TreeNode a geçirilir ve komşusuna $* atanır

4. stack taki peek edilen void pointer TreeNode ile cast edilip komşusuna onyum* atanır



CamScanner ile tarandı



CamScanner ile tarandı

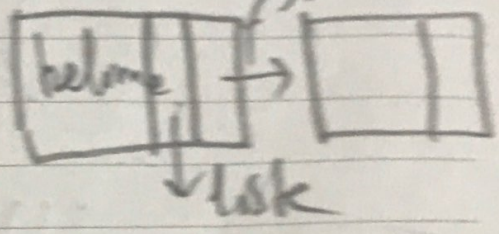
1) Kelime string bazlı mı olmalı

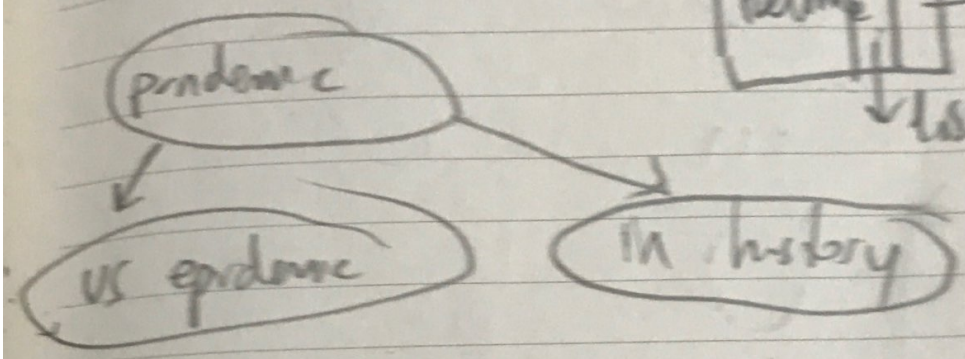
endemic → in definition

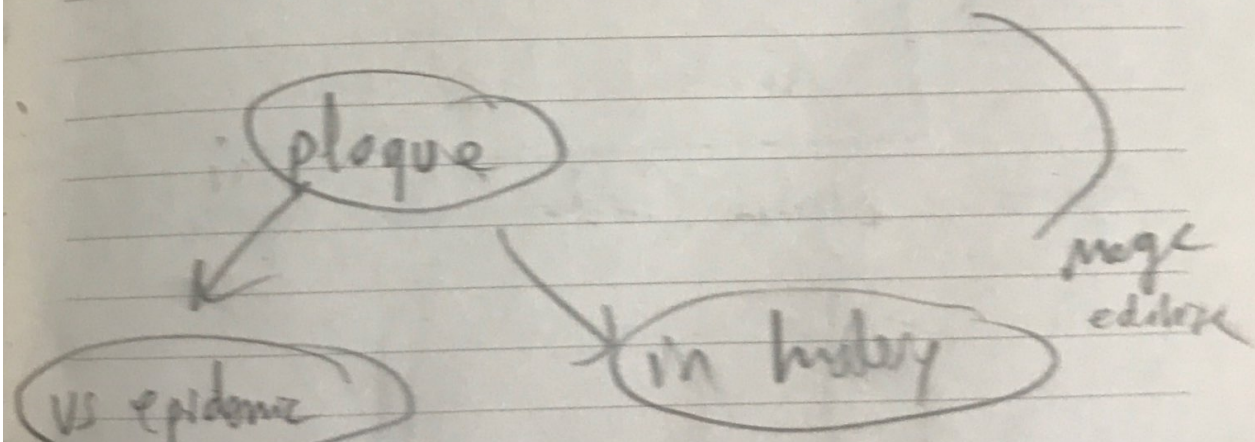
2) Arama bazlı mı olmalı

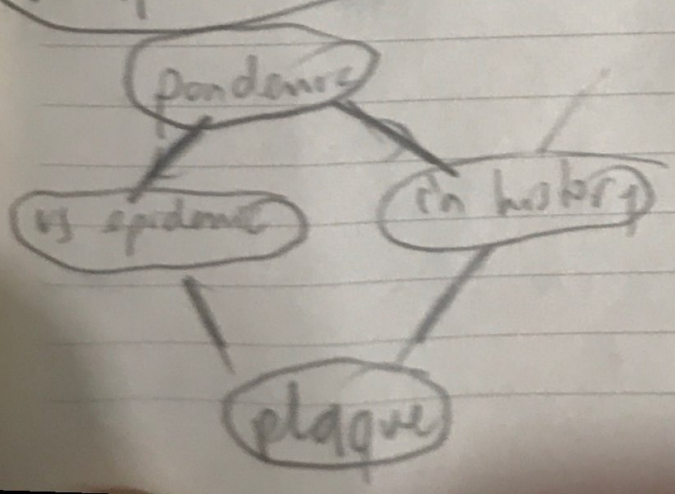
pandemic in definition

1. seviyede yapılırsa ortada ilişkilerle ilgili bir graf yapılabilir









seviyede olmalı