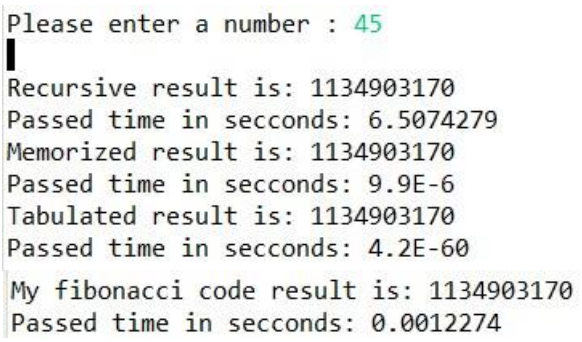

Dinamik programlama, karmaşık bir problemi alt problemlere parçalayarak o probleme çözüm bulmaya yarayan ve bunu yaparken de hafızadan faydalanan bir yöntemdir. Burada amaç yapılan yinelemeli işi ya daha az bir yinelemeyle ya da hiç yineleme olmadan hafızadan ödün vererek daha hızlı bir şekilde çözüme kavuşturmaktır.

Buna örnek olarak bir kümenin alt kümelerinde istenen parçayı bulmak ya da bir küme içinde iki eş toplamlı alt küme aramak söylenebilir.

Yani biz büyük sorunun eğer daha küçük alt parçalarıyla ilgileniyorsak dinamik programlama her zaman aklımızda bulunmalıdır.

Dinamik programlamanın birkaç çözüm yönteminden,problemden ve dinamik programlamaya dair terimlerden bahsedelim.

Memoization (Hatırlama): Hatırlama belli çözümler aradığımızda bunu normalden daha az yinelemeyle yapmayı sağlayan yöntemdir. Örneğin bir cevap için 10 kez recursive bir çağrıda bulunmak yerine belki de sadece sonuç için bunu yapacağımız işi hafızada tutmak kaydıyla 3,5 gibi çok daha az bir adıma indirger. Top-Down (Yukarıdan Aşağı) için kullanılır.

Tabulation (Tablolama): Eğer tüm çözümleri tarayarak bir sonuca varmamız gerekiyorsa,örneğin bir kümenin alt kümelerinin eleman toplamlarında girilen bir toplam değerini bulmak gibi,bunu defalarca kez yineler bir şekilde yapmak yerine bir tabloya kaydederek çözebilmek mümkün. Tablolama yineleme gerektirmeyen bir işlemdir. Bir adımın üzerinden tekrar geçmeyi gerektirmez. Bottom-Up (Aşağıdan Yukarı) için kullanılır.

Top-Down ve Bottom Up: Türkçe kelime toplulukları içinden bir örnekle gidelim. Eğer siz çözümünüz için özelden genele yani en küçük parçadan büyük parçaya ulaşır gibi ilerliyorsanız orada top-down var gibi düşünelim. Ama eğer ki biz problemin büyük parçasını küçük alt problemlere ayırarak istediğimiz sonuca erişmeye çalışıyorsak yani sanki genelden özele bir iş yapıyorsak burada bottom-up vardır. Top-

down için memoization kullanırız çünkü en küçük parçalarda yaptığımız işlemi bir hafızaya atarak bir üst sonuca ulaşırız ve her alt adımdaki sonucu hafızamızda tutarız. Bottom-upta ise büyük bir problemi küçük parçalara ayırır ve bir tabloya kaydederiz. Yani yinelemeli işlem yapmayız. Aslında var olanı parçalara böleriz. Ve bunu normal iş yüküne göre daha az sayıda işlem yaptığımız için tercih ederiz.

Çıkarımlar:

❁ Eğer bir problemin tüm parçalarıyla bir sonuca ulaşmamız gerekiyorsa tablolama-bottom up yöntemi daha açıklayıcı ve mantıklıdır.

❁ Eğer biz bir problemin belli bir parçasına ulaşmaya çalışıyorsak bunun için de memoization-top down yapmak daha karlı olur.

❁ Memoization-top down yinelemeliyken tablolama-bottom up yineleme içermez.

❁ Dinamik programlama çoğu problemimizde bize zamandan kar sağlarken hafızadan kaybımıza sebep olur. O nedenle dinamik programlamayı kullandığımız problemlerde parçaların hesabının aşırıya kaçması bir zaman sonra bize sağladığı karı faydasız bir duruma çevirebilir. Buna oluşturduğunuz tablonun 105*100 lük bir tablo olduğunu ve bize maliyetinin O(n*m) kaynaklı olduğunu düşünerek örnek verebiliriz.

References:

https://medium.com/@havvanurselamet/dinamik-programlama-nedir-y%C3%B6ntemleri-nelerdir-71258a1f8c62

https://www.geeksforgeeks.org/overlapping-subproblems-property-in-dynamic-programming-dp-1/

Channel name: Pepcoding

Playlist : Dynamic Programming

https://www.youtube.com/watch?v=A6mOASLl2Dg&list=PL-Jc9J83PIiG8fE6rj9F5a6uyQ5WPdqKy&index=2