به نام خدا

فاطمه مشهودی

4011928

رشته علوم کامپیوتر

دانشگاه دولتی فسا

پروژه پایانی درس ساختمان داده و الگوریتم ها با استاد خانم بخشنده

Sorting Algorithms

Merge sort

مرتب سازی ادغامی یکی از روش‌های مرتب‌سازی مبتنی بر تقسیم و غلبه (Divide & Conqure) است. مسئله را به زیر مسئله‌های کوچکتر تبدیل می‌کند و سپس آن‌را حل می‌کند. فرض کنید یک آرایه‌ی n عضوی از عناصر را دراختیار داریم. این آرایه را به 2 قسمت مساوی تقسیم می‌کنیم. در این حالت 2 آرایه n/2 عضوی داریم.

مجددا هرکدام از این آرایه‌ها را به دو بخش تقسیم می‌کنیم و آنقدر این عمل را ادامه می‌دهیم تا به آرایه‌های تک عنصری برسیم. در هنگام تقسیم شدن آرایه به دو آرایه، عملیات ادغام انجام شده و هرنیمه مرتب می‌شود و درنهایت یک آرایه مرتب‌شده از عناصر را دراختیار داریم. مرتبه اجرایی مرتب سازی ادغامی از O(n logn) است.

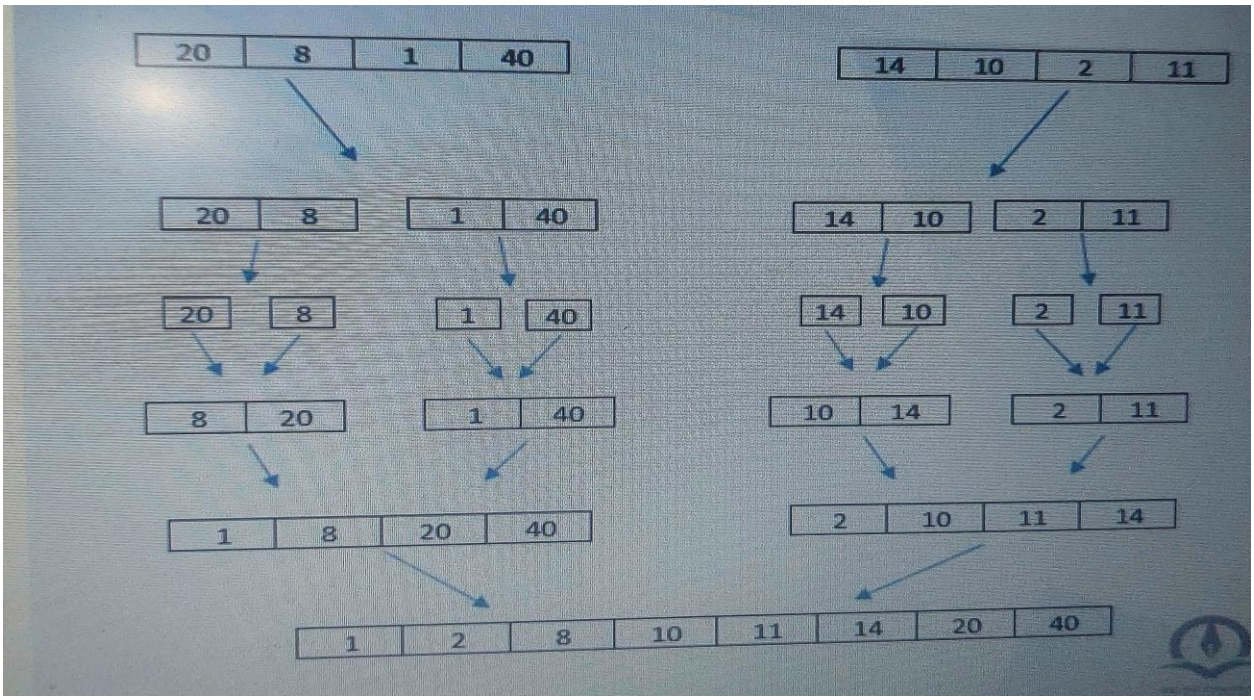
همان‌طور که اشاره شد، این روش مرتب‌سازی برگرفته از **تقسیم و غلبه** است که به صورت **بازگشتی** برروی آرایه اعمال می‌شود. برای توضیح این الگوریتم، درابتدا

مثالی را بررسی می‌کنیم که با این الگوریتم آشنایی پیدا کنیم.

فرض کنید آرایه از عناصر به شکل زیر داریم:

20	8	1	40	14	10	2	11

حال می‌خواهیم اعداد درون این آرایه را مرتب کنیم.
همان‌طور که گفته شده درابتدا آرایه را به دو
آرایه n/2عضوی تقسیم می‌کنیم:



همان‌طور که مشاهده می‌شود، توانستیم به روش **بازگشتی** آرایه را به 8 آرایه تک عضوی تقسیم کنیم و سپس هر دو آرایه تک عضوی را با یکدیگر مقایسه می‌کنیم و آرایه‌های 2 عضوی ایجاد می‌کنیم، این‌کار تا زمانی که یک آرایه 8 عنصری داشته باشیم، ادامه می‌یابد.

در فایل مرتب سازی ادغامی یک برای دیتاست 100 تا 1000 نوشته ایم و در فایل دوم برای دیتای 10 هزار تا صد هزار و در فایل سوم برای آرایه ای با اندازه ی یک میلیون تا ده میلیون طراحی کرده ایم.

دستورات iostream و vector و cstdlib و ctime به ترتیب کتابخانه های موردنیاز برای استفاده از ورودی و خروجی استاندارد و آرایه های داینامیک و اعداد تصادفی و زمان را فراهم می کنند.

در تابع merge دو زیرآرایه از آرایه اصلی arr ساخته می شود: L برای زیر آرایه از left تا mid و R برای زیرآرایه از .mid+1 to right

داده ها از arr به L,R کپی میشوند و سپس دو زیرآرایه با هم ادغام می شوند و نتیجه در []arr ذخیره می شود.

تابع merge sort الگوریتم را پیاده سازی می کند. ابتدا آرایه را به دو نیمه تقسیم می کند سپس هر یک از نیمه ها را مرتب سازی می کند و در نهایت آن ها را با هم ادغام می کند.

شرط توقف اجرای تابع زمانی است که left >= right باشدیعنی آرایه دارای یک عنصر یا هیچ عنصری نباشد.

در main یک آرایه با 1000 عنصر (بعنوان مثال در فایل اول مربوط) تعریف می شود و سپس مقداردهی تصادفی به عناصر آن انجام می شود.

آرایه قبل از مرتب سازی چاپ شده و سپس تابع mergesort بر روی آرایه فراخوانی می شود. پس از مرتب سازی آرایه بعد از مرتب سازی نمایش داده می شود.

در اینجا همچنین یک ثابت به نام size تعریف شده که مقدار آن برابر با 1000 است.(مثلا در فایل اول) این

ثابت برای اندازه آرایه مورد استفاده در این برنامه استفاده می شود.

خط بعدی آن از نوع int با نام arr تعریف شده است که دارای size عنصر است. این آرایه برای نگهداری اعداد قرار داده شده است.

در خط 29 تا 32 حلقه ی for داریک که این حلقه برای مقداردهی تصادفی به اعضای آرایه arr استفاده می شود.

لاین srand time 0 باعث می شود که تابع رندوم اعداد تصادفی متفاوت تولید کند.

عدد تصادفی بین 0 تا 9999 تولید می شود.

خط 35 تا38 آرایه arr را قبل از اعمال مرتب سازی نمایش می دهد.اینجا اعداد تصادفی که در مرحله ی قبلی تولید شده اند چاپ می شوند.

تابع mergesort بر روی آرایه arr فراخوانی می شود تا اعمال مرتب سازی را انجام دهد.

با اینکه arr.size()-1 حداکثر آخرین عنصر آرایه است که برای مرتب سازی استفاده می شود.

آرایه arr پس از اعمال مرتب سازی مجددا چاپ می شود تا تغییراتی که اعمال شده است نمایش داده شود.

از آرایه arr نقطه شروع left و میانه mid و نقطه پایان right زیرآرایه ای است که ادغام کرده ایم.

N1,n2 اندازه زیرآرایه های l,r را نشان می دهند.

Radix sort

مرتب‌سازی پایه‌ای یا **مرتب‌سازی مبنایی** (به انگلیسی: Radix sort) الگوریتمی است که لیستی با اندازهٔ ثابت و اعضایی با طول k را در زمان O(kn) انجام می‌دهد. ورودی‌ها را به بخش‌های کوچکی تقسیم می‌کنیم (اگر یک کلمه است آن را به حرف‌هایش

می‌شکنیم و اگر عدد است آن را به ارقامش) سپس ابتدا لیست را بر اساس کم ارزش‌ترین بیت (حرف یا رقم) مرتب می‌کنیم، سپس بر اساس دومین بیت، تا در نهایت بر اساس پرارزش‌ترین بیت. به این ترتیب پس از k مرحله لیست مرتب می‌شود. این روش مرتب‌سازی پایدار است و در تهیهٔ واژه نامه‌ها و مرتب‌سازی اعداد استفاده می‌شود.

معمولاً اعداد صحیحی که با الگوریتم‌های مرتب‌سازی پردازش می‌شوند را «کلیدها» می‌گویند، که می‌توانند به تنهایی موجود باشند یا همراه داده‌های دیگر.
مرتب‌سازی‌های مبنایی کم ارزش‌ترین رقم معمولاً اینگونه مرتب می‌کنند: کلیدهای کوتاه قبل از کلیدهای بلندتر می‌آید و کلیدهای هم طول هم به صورت لغت نامه‌ای مرتب می‌شوند. این با ترتیب معمولی اعداد صحیح منطبق است. مثل ترتیب: ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹، ۱۰. مرتب‌سازی‌های مبنایی پرارزش‌ترین رقم ترتیب لغت نامه‌ای دارند که برای مرتب کردن رشته‌ها مناسب است. مثل کلمات یا اعداد صحیح با طول ثابت. یک ترتیب مثلb«،c،d،e،f،g،h،i،j،» baوقتی

لغت نامه‌ای مرتب شود به صورت«b،ba،c،d،e،f،g،h،i،» jدر می‌آید. اگر ترتیب لغت نامه‌ای برای اعداد صحیح با طول متغیر اعمال شود، آنگاه نمایش اعداد ۱ تا ۱۰ خروجی«۱، ۱۰، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹" را پیدا می‌کند؛ بنابراین در این حالت برای درست مرتب شدن اعداد باید با گذاشتن فاصله از سمت چپ، اعداد کوتاه تر را با اعداد بلندتر هم طول کرد.

یک مرتب‌سازی مبنایی کم ارزش‌ترین رقم یک الگوریتم مرتب‌سازی پایدار سریع است که می‌تواند برای مرتب‌سازی کلیدها به ترتیب الفبایی استفاده شود. کلیدها ممکن است یک رشته از حروف یا ارقام داده شده در یک "مبناً باشند. پردازش از کم ارزش‌ترین رقم (یعنی رقم سمت راست) شروع می‌شود و به پرارزش‌ترین رقم (رقم سمت چپ) می‌رسد. ترتیبی که رقم‌ها مورد پردازش قرار می‌گیرند در مرتب‌سازی مبنایی کم ارزش‌ترین رقم برعکس این ترتیب در مرتب‌سازی مبنایی پرارزش‌ترین رقم است.

یک مرتب‌سازی مبنایی کم ارزش‌ترین رقم در (O (nk کار می‌کند که n تعداد کلیدها و k میانگین طول کلیدهاست. برای رسیدن به این کارایی در مرتب‌سازی کلیدها با طول متغیر شما باید همه کلیدها با طول یکسان را در یک گروه قرار دهید و جداگانه یک مرتب‌سازی مبنایی کم ارزش‌ترین رقم را روی هر گروه از کلیدهای هم طول، از کوتاه‌ترین تا بلندترین اجرا کنید.

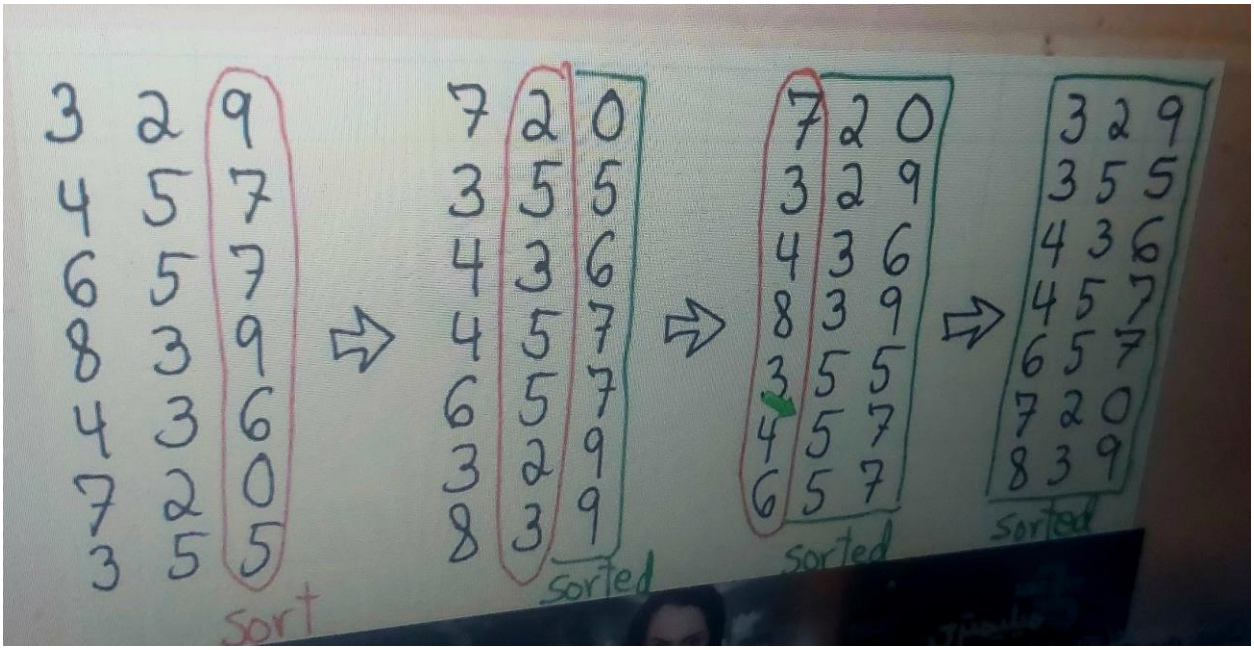
به عبارت دیگر:

۱ .کم ارزش‌ترین رقم کلیدها را برمی داریم.

۲ .کلیدها را براساس آن رقم گروه بندی می‌کنیم، ولی از طرف دیگر ترتیب اصلی کلیدها را حفظ می‌کنیم. (این باعث می‌شود مرتب‌سازی مبنایی کم ارزش‌ترین رقم، پایدار شود)

۳ .عملیات گروه بندی را با ارقام کم ارزش بعدی ادامه می‌دهیم.

در قدم ۲ مرتب‌سازی که استفاده می‌شود مرتب‌سازی پیمانه‌ای یا مرتب‌ساز شمارشی است که وقتی کارآمد هستند که تعداد کمی عدد داشته باشیم.



در فایل های این مرتب سازی تابع getmax برای یافتن بیشترین عدد در آرایه استفاده می شود. این مقدار برای برای تعیین تعداد دفعاتی که الگوریتم باید اجرا شود (بر اساس تعداد ارقام بیشترین عدد) استفاده می شود.

تابع radixsort مرتب سازی را بر روی آرایه انجام می دهد.

از exp به عنوان توان اعداد ده برای جداسازی رقم استفاده می شود.(واحد دهدهی و صددهی و...)

اعداد بر اساس رقم exp مرتب می شوند و در آرایه خروجی قرار می گیرند.

در هر مرحله count برای شمارش تعداد با توجه به رقم موردنظر استفاده می شود.

آرایه خروجی در نهایت به آرایه اصلی کپی می شود.

در تابع main یک آرایه با اندازه تصادفی مثلا 10000 تا 100000 ایجاد می شود و اعداد تصادفی در آن قرار می گیرند.

آرایه قبل از مرتب سازی چاپ می شود.

Radix sort اجرا می شود.

آرایه پس از مرتب سازی چاپ می شود.

همچنین با هر 100 عدد در هر خط چاپ می شود برای جلوگیری از چاپ طولانی.

اجرای دیتای یک میلیون ... و بزرگ تر از آن ممکن است ز مان بر باشد و نیاز به سرورهای با منابع قدرتمند دارد تا به درستی انجام شود.

هر دو الگوریتم برای مرتب سازی دیتای بزرگ مناسب هستند اما به دلایل مختلفی که در زیر توضیح داده می شود:

Merge sort

1. پیچیدگی زمانی : دارای پیچیدگی زمانی لگاریتمی است که از لحاظ زمانی بسیار کارآمد است. این الگوریتم می تواند آرایه های با اندازه های بسیار بزرگ را به خوبی مرتب کند.
2. مناسب برای حاقظه های پیچیده: نیازمند حافظه ی اضافی برای ادغام زیر آرایه ها است که در مواقعی که منابع حافظه کافی در دسترس است مناسب است.
3. پراکندگی داده ها : بهترین عملکرد را دارد زمانی که داده ها نسبتا پراکنده یا از نوع داده ای هستند که دسترسی به آنها بصورت تصادفی انجام می شود.

Radix sort

1. پیچیدگی زمانی : دارای پیچیدگی زمانی خطی است که برای برخی حالات خاص بسیار کارآمد

است.این الگوریتم بخصوص برای مرتب سازی اعداد صحیح مثبت مناسب است.

2. عملکرد بر اساس رقم : از ترتیب رقمی برای مرتب سازی استفاده می کند یعنی داده ها را بر اساس رقم یک به یک مرتب می کند. این ویژگی باعث می شود که در برخی حالات خاص مانند اعداد با تعداد ارقام ثابت بهترین عملکرد را داشته باشد.
3. مناسب برای داده های ثابت ارقام : برای داده هایی که ثابت ارقام هستند یا می توانند به ارقام ثابت تبدیل شوند بسیار مناسب است. بنابراین برای برخی حالات خاص از merge sort ممکن است بهترین عملکرد را داشته باشد.

نتیجه گیری :

انتخاب بین این دو الگوریتم بستگی به ویژگی های خاص داده ها دارد. اگر داده ها پراکنده و حافظه کافی باشد الگوریتم ادغامی بهترین گزینه است. اما اگر داده

ها از نوعی با ثابت ارقام هستند الگوریتم radix می تواند گزینه مناسبی باشد.

در کاربردهای واقعی ممکن است از الگوریتم های مختلف به صورت ترکیبی استفاده شود بسته به نیازهای خاص و مشخصات داده ها.

Shell sort

مرتب‌سازی شِل یا **مرتب سازی صدفی** یکی از قدیمی‌ترین الگوریتم‌های مرتب‌سازی و تعمیمی از مرتب‌سازی درجی با در نظر گرفتن دو نکته زیر است:

1. الگوریتم مرتب‌سازی درجی درصورتی کارآمد است که داده‌ها تقریباً مرتب باشند.
2. مرتب‌سازی درجی معمولاً کم بازده‌است چون مقادیر را در هر زمان فقط به اندازه یک موقعیت جابجا می‌کند.

مرتب‌سازی صدفی، الگوریتمی سریع و کارآمد و در عین حال یادگیری و پیاده‌سازی آن ساده است.

البته این نکته قابل توجه است که مرتب‌سازی صدفی درحقیقت به تنهایی داده‌ها را مرتب نمی‌کند بلکه به نوعی از سایر مرتب‌سازی‌ها استفاده کرده و با یک دسته‌بندی مناسب که موجب می‌شود تعداد دفعاتی که هر داده بررسی می‌شود کاهش یابد، کارایی آن‌ها را افزایش می‌دهد.

در روش مرتب‌سازی درجی در بدترین حالت الگوریتم از $O(n^2)$ است درحالی که با مرتب‌سازی صدفی این زمان به $O(n \log^2)$ کاهش یافته‌است.

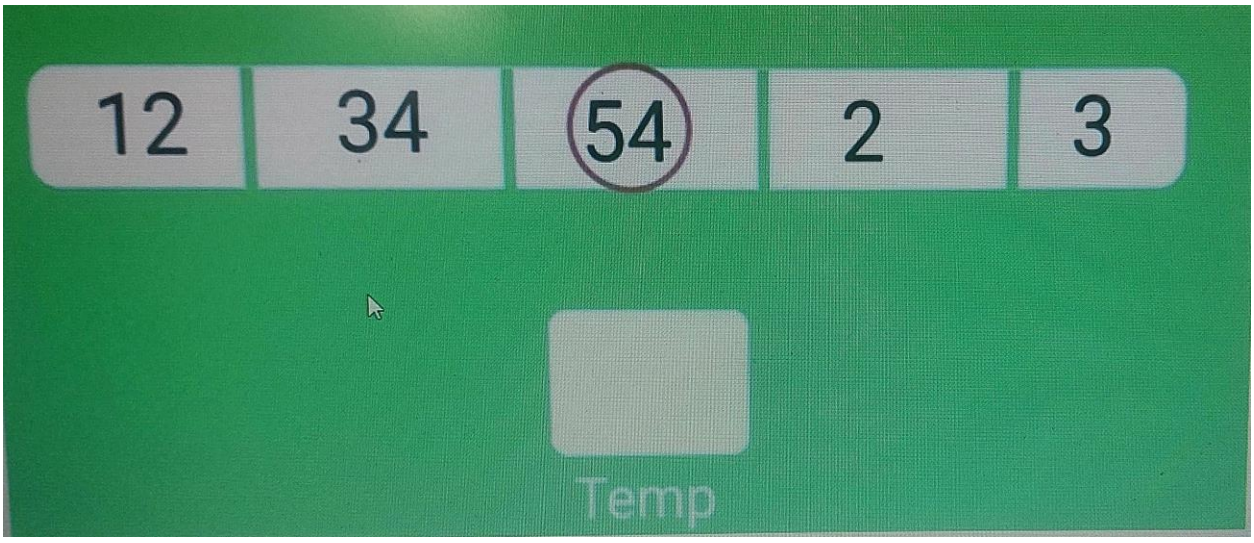
در مرتب‌سازی صدفی، ابتدا داده‌ها در گام‌های بزرگتری جابه جا می‌شوند اما در هر مرحله فاصله این جابه جایی‌ها کم تر شده و به جابه جایی‌های جزئی می‌رسد.

«مرتب سازی شل (Shell Sort)» یکی از انواع «مرتب‌سازی درجی (Insertion Sort)» است.
در مرتب‌سازی درجی، عناصر تنها یک موقعیت، رو به

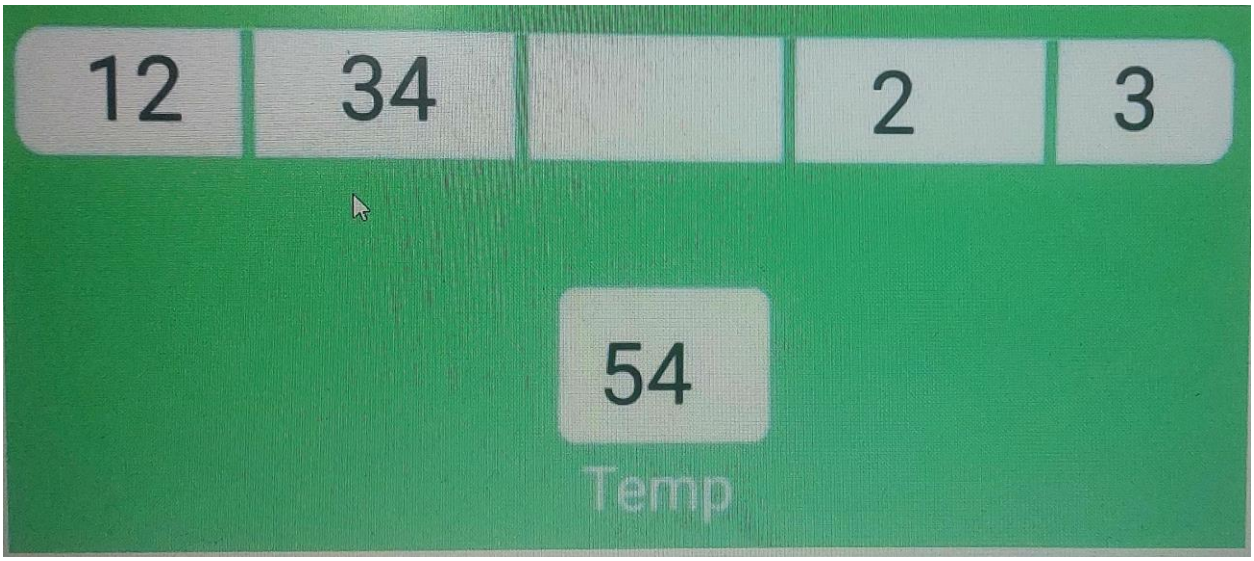
جلو جا به جا می‌شوند. هنگامی که یک عنصر به تعداد زیادی خانه به جلوتر جا به جا می‌شود، حرکات زیادی باید انجام شود.

ایده مرتب سازی شل آن است که امکان جا به جایی عناصر دور را فراهم کند. در مرتب سازی شل، آرایه برای مقادیر بزرگh ، به صورت h-sort مرتب‌سازی می‌شود. کاهش مقدار h تا هنگامی که ۱ شود، ادامه خواهد داشت. یک آرایه h-sorted گفته می‌شود اگر همه زیرلیست‌هایh اُمین عنصر، مرتب‌سازی شده باشند.

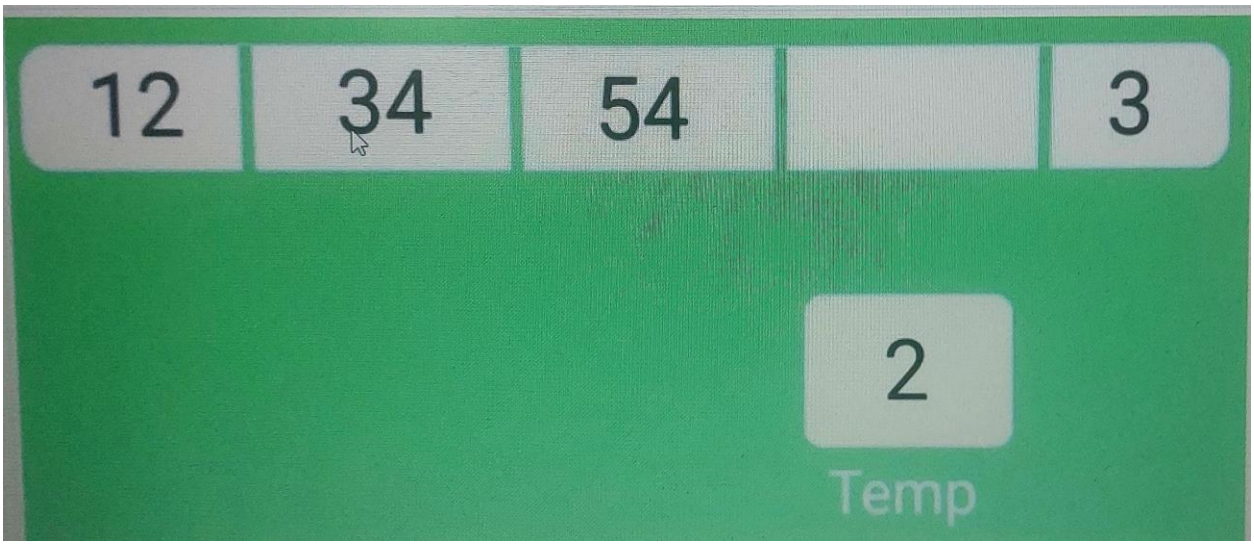
کار از $gap=n/2$آغاز می‌شود (در این مثال، ۲ است).



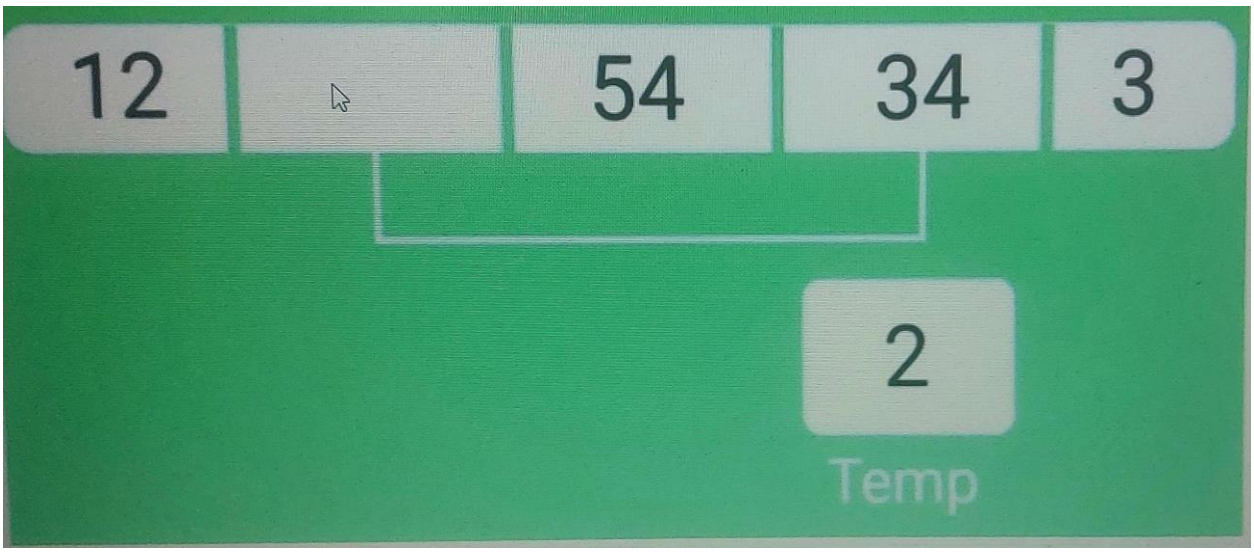
۵۴ را به Temp منتقل کن.



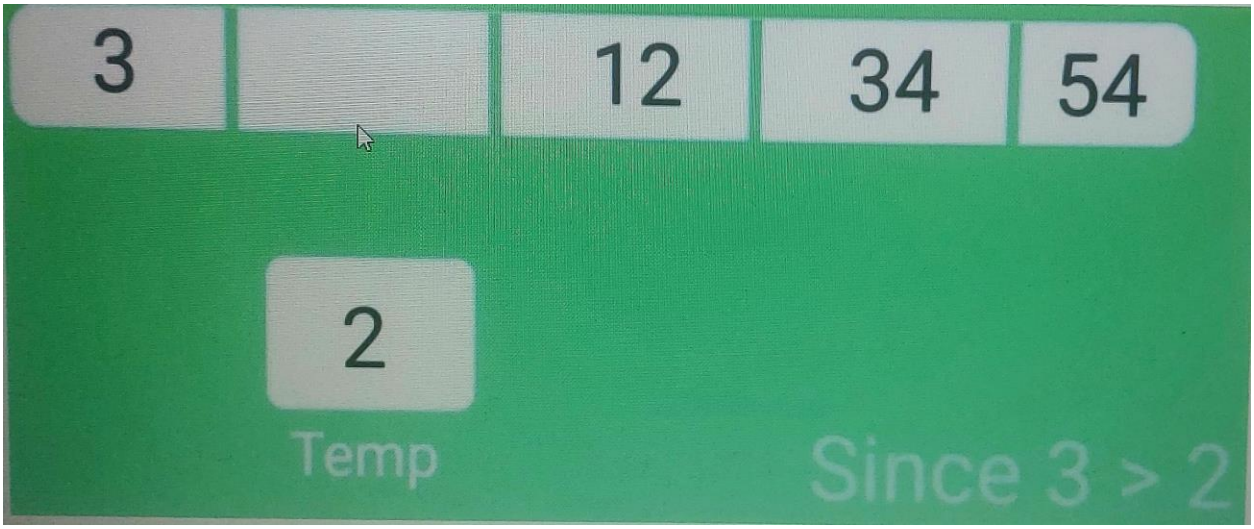
عناصر سمت چپ ۵۴، کوچک‌تر از آن هستند و بنابراین، نیازی به تغییر نیست. لذا ۵۴ را به جای خود بازگردان. اکنون باید یکی یکی عناصر سمت راست شکاف را انتخاب کرد و آن‌ها را در موقعیت مناسب قرار داد.



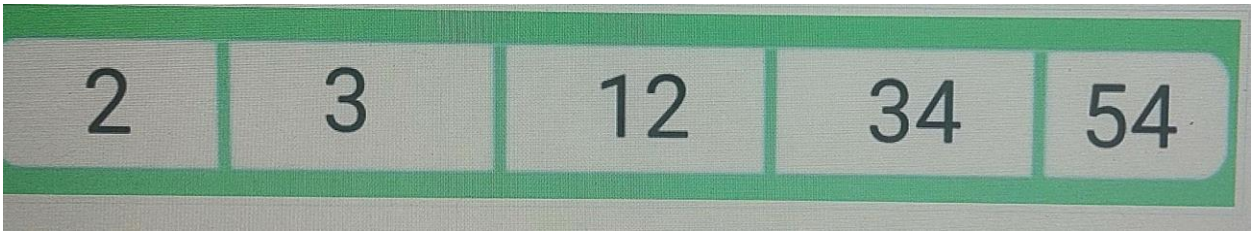
ابتدا ۲ را به Temp ببر. ۲ را با arr[3-2]=34 مقایسه و به arr[gap+1 = 3] جا به جا کن.



اکنون، شکاف به $1(n/4)$کاهش پیدا می‌کند. همه عناصر را با شروع از arr[1] انتخاب و آن‌ها را عناصر در فاصله شکاف، مقایسه کن.



اکنون شکاف به ۰ می‌رسد. مرتب‌سازی متوقف می‌شود و حاصل، یک آرایه مرتب شده است.



تابع اصلی shell sort که vector int &arr یک ورودی می گیرد و آن را مرتب می کند.

در این تابع یک حلقه برای محاسبه دنباله گام اجرا می شود.این دنباله از اعداد n/2 شروع می شود و به صورت پیوسته کاهش می یابد تا به یک برسد.

سپس برای هرگام insertion sort بر روی زیر آرایه های با فاصله gap اجرا می شود.

Main تابع اصلی برنامه است که آرایه عنصری را ایجاد می کند و مقداردهی تصادفی به آن انجام می شود.

شل یک الگوریتم مرتب سازی با کارایی خوب برای آرایه های با اندازه های متوسط است و می تواند برای آرایه های با حجم تا حدود 100000 عنصر مناسب باشد. با افزایش آرایه به بیش از این ممکن است این الگوریتم عملکرد کمتری داشته باشد و بهینه سازی های دیگر مانند quick , merge ممکن است بهتر باشند.

----------پایان----------