نحوه ی کار کلی الگوریتم:

این الگوریتم ابتدا جدول ۹*۹ را در آرایه ی دوبعدی ۹ در ۹ ذخیره میکند و اول مقدار دهی اش میکند، به این صورت که در هر زیرمربع اعداد غیرتکراری ۱ تا ۹ را قرار میدهد. سپس هزینه ی جدول را محاسبه میکند به این صورت که اعداد تکراری در هر سطر و ستون و زیرمربع هارا میشمارد و تعداد error های جدول را اصطلاحا محاسبه میکند.

پس از آن الگوریتم با جابجا کردن دو خانه در هر زیرمربع از یک جدول فرزند میسازد و از بین آنها یکی انتخاب میشود و هزینه اش مجاسبه میشود. اگر هزینه اش از هزینه ی پدر کمتر بود امتخاب میشود و اگر بیشتر بود ابتدا با یک تابع acceptance مقدار فرمول sa را محاسبه میکند و عدد رندمی تولید میکند و با آن مقایسه میکند و اگر بیشتر از آن بود انتخاب این فرزند قطعی میشود و اگر کمتر بود قطعی نمیشود و فرزند دیگری را انتخاب میکند. هنگام قطعی شدن یک فرزند آنرا به عنوان جدول اصلی در نظر میگیریم و در تکرار های دیگر الگوریتم از آن جدول استفاده میکنیم.

الگوریتم تا جایی که برایش تعیین کردیم تکرار میشود که در اینجا ۲۰ بار در نظر گرفته شده. در هر بار اجرا خروجی را به عنوان یک جدول و هزینه ی آن میدهد.

جزئیات الگوریتم:

این الگوریتم از ۵ تابع تشکیل شده است،

تابع initialize که آرایه ی خالی را و یا جدول خالی را با اعداد تصادفی پر میکند. بدین صورت کار میکند که به ازای هر سطر و ستون و زیرمربع جدول حلقه داریم و داخل هر یک از خانه ها برای آن خانه یک عدد رندم از ۰ تا ۹ انتخاب میکند و چک میکند که برابر ۰ نباشد و سپس چک میکند که آن عدد انتخابی در آن زیرمربع تکراری نباشد. اگر در زیرمربع تکراری بود یک عدد دیگر انتخاب کرده و از اول چک میکند. اگر هم نبود به عنوان مقدار آن خانه قرار داده میشود.

مقادیر p و q برای چک کردن در زیرمربعی است که در آن هستیم و هربار که برای یک خانه عدد اشتباهی انتخاب کرده باشیم این متغیر ها از اول مقداردهی میشوند که از اول زیرمربع را برای مقدار انتخابی جدید چک کنند.

متغیر بولین isValid برای این است که به ازای هر مقداری که تابع رندم انتخاب میکند در آخر که همه ی زیرمربع را چک کردیم مقدار isValid را false قرار میدهیم اگر آن عدد در زیرمربع تکراری باشد و true اگر نباشد و اگر isValid برابر true بود آن مقدار برای آن خانه قرار داده میشود.

تابع cost به ازای هر سطر و ستون و زیرمربع جدولی که به عنوان ورودی میبیند اگر عدد تکراری در هریک از آنها پیدا کرد متغیر sscore را یکی بالا میبرد و در آخر این متغیر را به عنوان مجموع error هایی که جدولمان دارد یا هزینه ای که دارد برمیگرداند.

تابع chooseChild یک فرزند برای جدول اصلی مان انتخاب میکند. ابتدا جدول اصلی را در یک جدول فرزند childPuzzle کپی میکند که تغییرات را روی آن انجام دهد و اگر قطعی نشد جدول پدر را همچنان داشته باشیم. سپس در هر زیرمربع دو سطر و دو ستون را به طور تصادفی انتخاب میکند و اگر با هم برابر نبودند (یعنی یک خانه را دوبار انتخاب نکرده بودیم) مقادیر موجود در خانه ها را باهم عوض میکنند.

تابع acceptance یک temperature برای کد انتخاب میکند و با توجه به فرمول SA تفاوت میان هزینه ی پدر و هزینه ی فرزند(که به عنوان ورودی میگیردشان)را تقسیم بر temperature میکند و سپس یک عدد رندم تولید میکند و آنرا با e به توان فرمولی که گفته شد مقایسه میکند و اگر از آن کمتر بود true و اگر بیشتر بود false برمیگرداند که به معنی قطعی شدن یا نشدن آن فرزند با آن هزینه ی مورد نظر است.

تابع print هم که به ترتیب از اول آرایه ی دوبعدی شروع به چاپ جدول میکند و سه تا سه تا خط میکذارد که زیرمربع ها مشخص شوند.

در main ابتدا تعداد دفعات تکرار الگوریتم را مشخص میکنیم و سپس جدول را مقداردهی اولیه میکنیم وهزینه ی جدول اصلی را محاسب میکنیم. بعد فرزند تولید کرده و هزینه ی آنرا با هزینه ی جدول اصلی مقایسه میکنیم و اگر کمتر بود جدول ها جابه جا و انتخاب قطعی میشود و اگر بیشتر بود تابع acceptance را صدا میکنیم و اگر trueبرگرداند پس انتخاب میشود و اگر نه فرزند دیگری انتخاب میشود.

کل این اعمال در یک while است که به تعداد دفعات مورد نظر ما تکرار میشود.